SPL-1 Project Report

Movie Ticket Booking System

Submitted by

Nandan Bhowmick

BSSE Roll No. : 1436

BSSE Session: 2021-2022

Submitted to

Dr. Md. Shariful Islam

Professor

Institute of Information Technology

University of Dhaka

Signature of Supervisor:..........................

Date:......................................

Institute of Information Technology

University of Dhaka

Submission date: 17-12-2023

Table of Contents

1) Introduction

Movie Ticket Booking System is a comprehensive C++ application designed to streamline the movie-watching experience for users. This terminal-based application allows users to effortlessly browse available movies, check showtimes, and reserve tickets in real-time. Additionally, the system includes a recommendation feature that suggests movies based on user preferences and viewing history. With its simplicity and user-friendly design, the system aims to enhance the movie-watching experience by providing a hassle-free approach to discovering and booking movies. It can be divided into 3 main parts:

I. Login System:
   - Implement a secure login system requiring valid credentials (username/email and password), ensuring user data privacy and security.
   - Allow new users to create accounts by providing necessary information.

II. Booking and Cancellation:
   - Display comprehensive information about available movies, including title, genre, showtimes, and halls.
   - Enable users to select movies, choose showtimes, and reserve seats. Seats availability to be updated at the same time.
   - Provide a mechanism for users to cancel booked tickets, including a time restriction.
   - Allow users to view their booking history, including past and present reservations.

III. Recommendation system:
   - A simple recommendation system that analyzes the user's history and recommends movies.

2) Background of the Project

The Movie Ticket Booking and Recommendation System emerged from the need to streamline and enhance the movie-watching experience for users. In a world where technology plays a pivotal role in simplifying everyday tasks, the project addresses the complexities associated with booking movie tickets and aims to offer users a seamless and personalized movie-going journey.

The project was conceived with several objectives in mind:

I. User Convenience:
   - The traditional process of purchasing movie tickets often involves long queues and manual efforts. The project seeks to provide users with a convenient and efficient alternative, allowing them to book tickets and manage their bookings from the comfort of their homes.

II. Data Management:
   - Storing and managing user information, movie details, and booking records is a critical aspect of the project. Leveraging file handling in C++, the system ensures secure and organized data storage, retrieval, and modification.

III. Personalized Recommendations:
   - Recognizing the importance of personalized user experiences, the project incorporates a recommendation system. By analyzing user booking history, the

system aims to recommend movies tailored to individual preferences, encouraging users to explore a variety of genres.

IV. Accessible Technology:
   - The decision to implement a terminal-based interface reflects a commitment to accessibility. The project caters to users with varying levels of technical expertise, providing an interface that is easy to navigate and interact with.

In short, the Movie Ticket Booking and Recommendation System is not just a technological solution; it is a response to the evolving expectations and demands of modern consumers. By combining user-friendly technology with efficient data management and personalized recommendations, the project strives to enhance the overall movie-going experience for users.

3) Description of the Project

A detailed description of the key features and their functionalities is given below.

I. Login System: If the user is using the program for the first time, they will have to sign up to create a new account with necessary information. The email the user enters has to be unique as it will be used to uniquely identify the user. The password will be hashed to securely save it in the file of that user. If the user already has an account, they will simply log in with necessary credentials(email and password). The user will be given at most 3 chances to successfully log in, otherwise the program will be exited. After signing up/logging in, the user will be taken to the User Menu, where the user will have the options to book a ticket, cancel a booking, view history, check what food and drinks are available in the theater and log out.

II. Booking a ticket: The movie schedule will be displayed for the user to choose which show they want to watch. They may choose to see the details of each available movie to get an idea about what they are about. Then they will be asked to enter a serial number and a time slot for the movie they want to watch. After that, they will be taken to a new screen where the seats available will be displayed(as arranged in the theater). The user will choose desired seats(maximum 5). After that they will be asked to pay the price for their booking. After entering the correct amount, booking is successful and the details of the booking are displayed on screen. The booking information will be stored in a separate file of "booking list" as well as in the user history. With the completion of the booking, the user is taken back to the User Menu.

III. Recommendation System: During booking a ticket, when the movie schedule is displayed on the screen for the user to choose the movie they want to watch, a list of movies from the available movies are recommended to the user based on their history. The history of the user is analyzed and the frequency of genres of movies that they watched are counted, and then available movies with the top two genres are recommended.

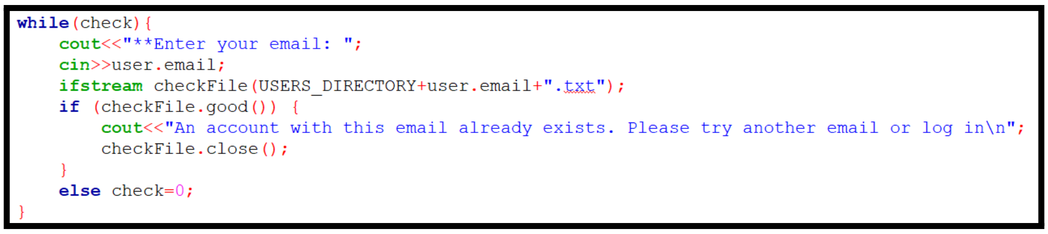
IV. Canceling a booking: The history of the user will be displayed on screen for the user to choose which booking they want to cancel. If the booking is being attempted to be canceled at least 5 hours before the showtime, the booking will be canceled with full refund. Otherwise, the booking cannot be canceled. With a successful booking, the booking information is removed from the booking list as well as from the user history.

V. Extra feature: The extra feature is simply to check out what food and drinks are available in the theater. A table of items and their respective prices will be displayed.

VI. Log out: If the user logs out, they will be taken back to the home window, i.e. the login/sign up window, from which they may choose to exit the program.

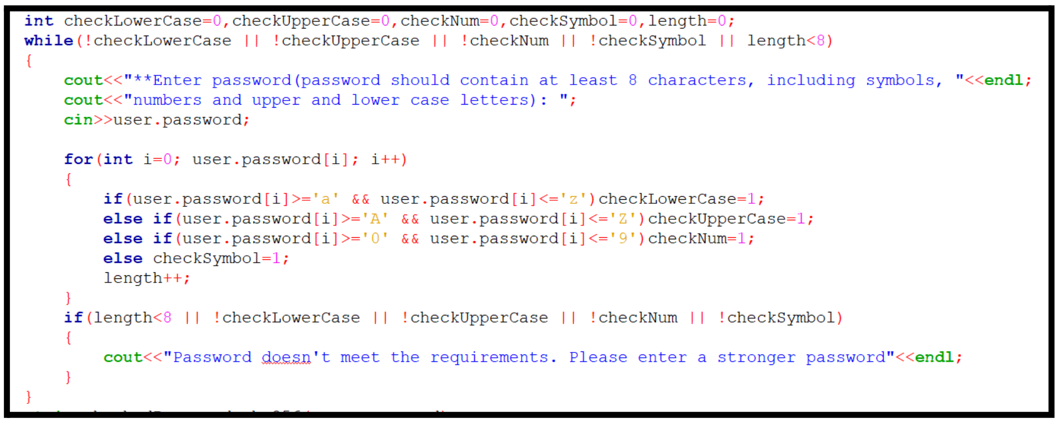
4) Implementation

- Login System:
  - ➢ Sign up: A structure is made that contains the information needed for the new account during sign up; this helps reduce complexity. It is ensured that the email entered is unique and no other users have the same email. The password is ensured to have at least 8 characters, with uppercase letters, lowercase letters, numbers and symbols. The password is hashed to enhance security. This required the addition of the OpenSSL library to the project.



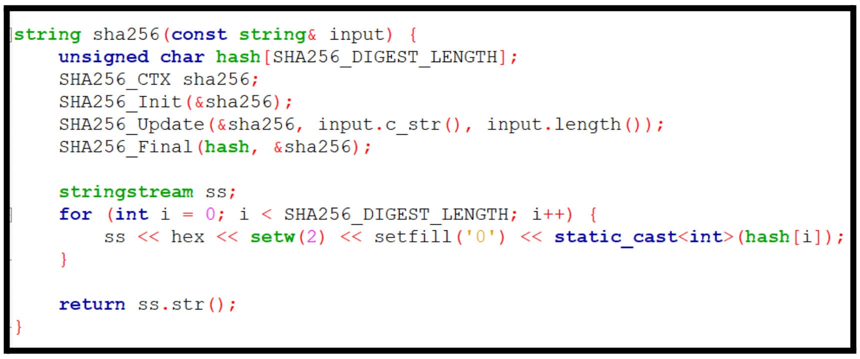
```
while(check){
    cout<<"**Enter your email: ";
    cin>>user.email;
    ifstream checkFile(USERS_DIRECTORY+user.email+".txt");
    if (checkFile.good()) {
        cout<<"An account with this email already exists. Please try another email or log in\n";
        checkFile.close();
    }
    else check=0;
}
```

Fig-1: Setting up unique email during sign up



```
int checkLowerCase=0,checkUpperCase=0,checkNum=0,checkSymbol=0,length=0;
while(!checkLowerCase || !checkUpperCase || !checkNum || !checkSymbol || length<8)
{
    cout<<"**Enter password(password should contain at least 8 characters, including symbols, "<<endl;
    cout<<"numbers and upper and lower case letters): ";
    cin>>user.password;

    for(int i=0; user.password[i]; i++)
    {
        if(user.password[i]>='a' && user.password[i]<='z')checkLowerCase=1;
        else if(user.password[i]>='A' && user.password[i]<='Z')checkUpperCase=1;
        else if(user.password[i]>='0' && user.password[i]<='9')checkNum=1;
        else checkSymbol=1;
        length++;
    }
    if(length<8 || !checkLowerCase || !checkUpperCase || !checkNum || !checkSymbol)
    {
        cout<<"Password doesn't meet the requirements. Please enter a stronger password"<<endl;
    }
}
```

Fig-2: Setting up a password

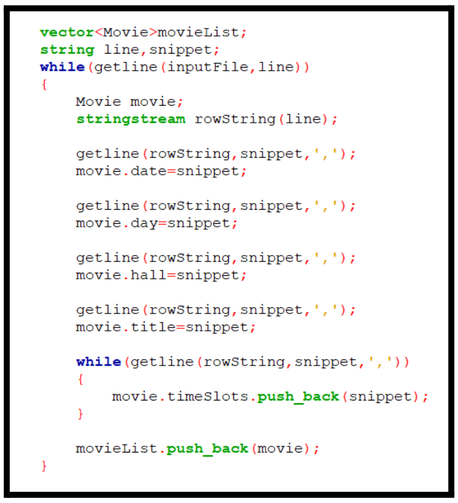


```
string sha256(const string& input) {
    unsigned char hash[SHA256_DIGEST_LENGTH];
    SHA256_CTX sha256;
    SHA256_Init(&sha256);
    SHA256_Update(&sha256, input.c_str(), input.length());
    SHA256_Final(hash, &sha256);

    stringstream ss;
    for (int i = 0; i < SHA256_DIGEST_LENGTH; i++) {
        ss << hex << setw(2) << setfill('0') << static_cast<int>(hash[i]);
    }

    return ss.str();
}
```

Fig-3: Implementing SHA-256 for password hashing

  - ➢ Login: The login function is quite simple- ask the user for email and password, hash the password using the same sha256() function and check if such a user exists. Maximum three chances will be given, which is done using a loop.

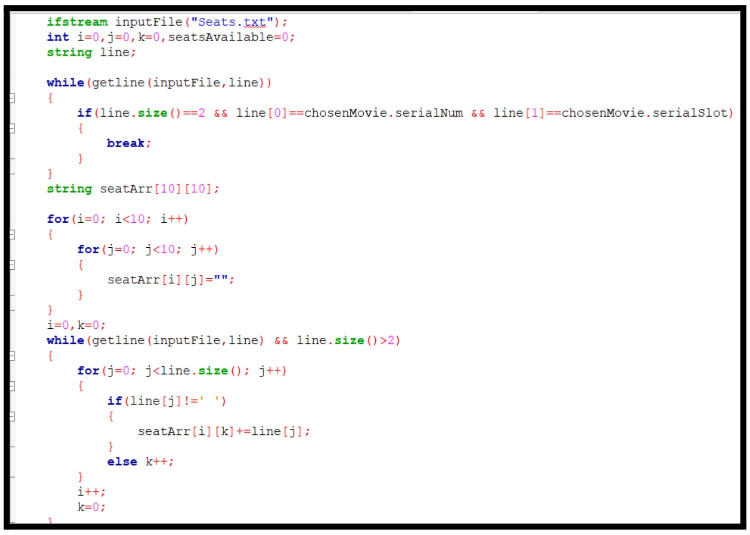
- Booking ticket:
  - ➢ The movie schedule is first displayed on screen for the user to choose the serial number and time slot of desired movie. The schedule is stored in a file from which it is read and displayed. The screenshot below shows how all the individual information about a movie is being read from a file and stored in a structure 'movie', which is then stored in a vector of that structure type.



```
vector<Movie>movieList;
string line,snippet;
while(getline(inputFile,line))
{
    Movie movie;
    stringstream rowString(line);

    getline(rowString,snippet,',');
    movie.date=snippet;

    getline(rowString,snippet,',');
    movie.day=snippet;

    getline(rowString,snippet,',');
    movie.hall=snippet;

    getline(rowString,snippet,',');
    movie.title=snippet;

    while(getline(rowString,snippet,','))
    {
        movie.timeSlots.push_back(snippet);
    }

    movieList.push_back(movie);
}
```

Fig-4: Reading movie schedule from file

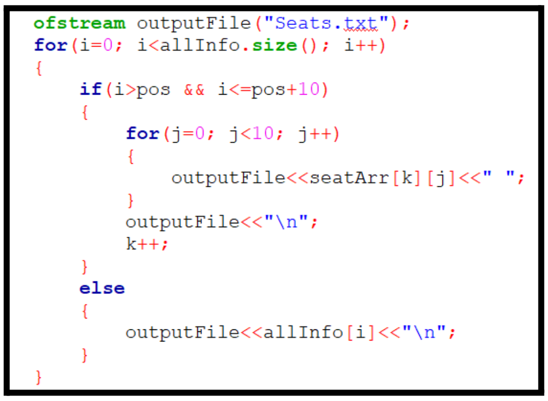
After taking the serial number and time slot number from user, the seat chart for that showtime is read from another file. The screenshot is shown below:



```
ifstream inputFile("Seats.txt");
int i=0,j=0,k=0,seatsAvailable=0;
string line;

while(getline(inputFile,line))
{
    if(line.size()==2 && line[0]==chosenMovie.serialNum && line[1]==chosenMovie.serialSlot)
    {
        break;
    }
}
string seatArr[10][10];

for(i=0; i<10; i++)
{
    for(j=0; j<10; j++)
    {
        seatArr[i][j]="";
    }
}
i=0,k=0;
while(getline(inputFile,line) && line.size()>2)
{
    for(j=0; j<line.size(); j++)
    {
        if(line[j]!=' ')
        {
            seatArr[i][k]+=line[j];
        }
        else k++;
    }
    i++;
    k=0;
}
```

Fig-5: Reading seat chart from file

The user then enters the seats of their choice(maximum 5) from the chart on display. The booked seats are then marked with a 'X', and the file containing the seat chart is updated. The screenshot below shows this part:



```
ofstream outputFile("Seats.txt");
for(i=0; i<allInfo.size(); i++)
{
    if(i>pos && i<=pos+10)
    {
        for(j=0; j<10; j++)
        {
            outputFile<<seatArr[k][j]<<" ";
        }
        outputFile<<"\n";
        k++;
    }
    else
    {
        outputFile<<allInfo[i]<<"\n";
    }
}
```

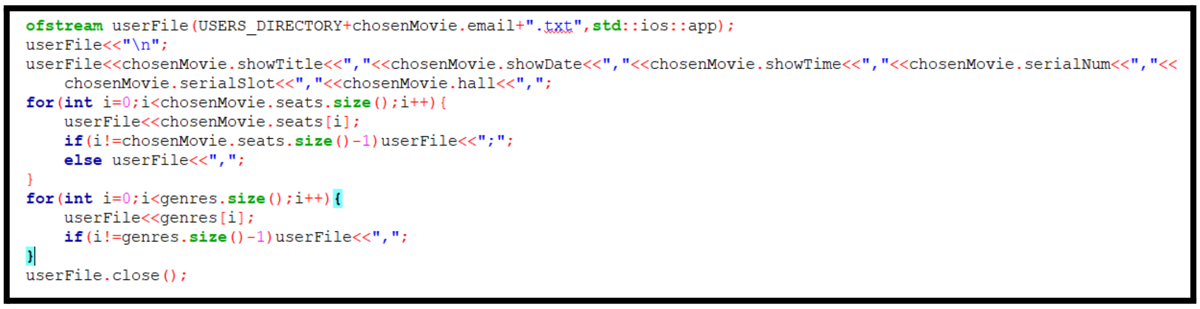
Fig-6: Updating seat availability chart

The 'allInfo' vector contains the seat numbers of all shows. When position 'pos for

the selected serial number and slot is found, an inner loop is used to update that specific showtime seats.

The booking information is also added to the history of the user file.

The updating of user history is shown in the screenshot below:

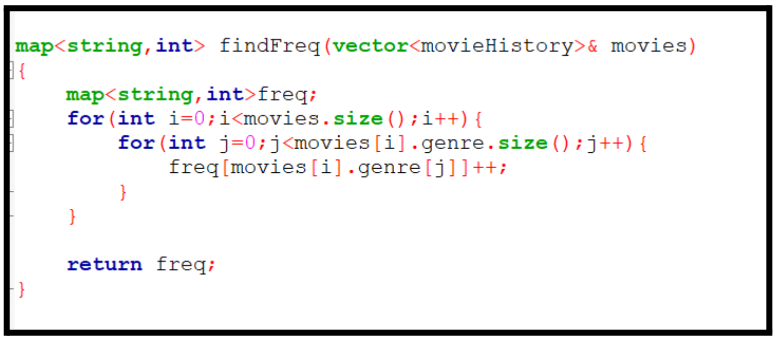


```
ofstream userFile(USERS_DIRECTORY+chosenMovie.email+".txt",std::ios::app);
userFile<<"\n";
userFile<<chosenMovie.showTitle<<","<<chosenMovie.showDate<<","<<chosenMovie.showTime<<","<<chosenMovie.serialNum<<","<<
    chosenMovie.serialSlot<<","<<chosenMovie.hall<<",";
for(int i=0;i<chosenMovie.seats.size();i++){
    userFile<<chosenMovie.seats[i];
    if(i!=chosenMovie.seats.size()-1)userFile<<";";
    else userFile<<",";
}
for(int i=0;i<genres.size();i++){
    userFile<<genres[i];
    if(i!=genres.size()-1)userFile<<",";
}
userFile.close();
```

Fig-7: Adding new booking information to user history

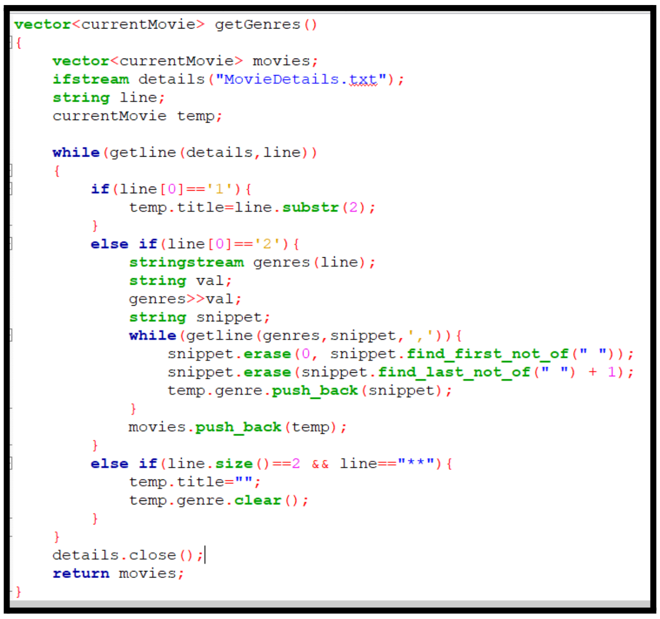
The booking information is printed after the user enters the amount of the ticket price.

- Recommendation System:
    - ➢ For recommending movies to the user, the history of the user is first read. Then the frequency of the genres of movies watched is counted with the help of a map. The genres of available movies are then found and then movies with the 2 most frequent genres are stored in a set and printed below the movie schedule during booking.



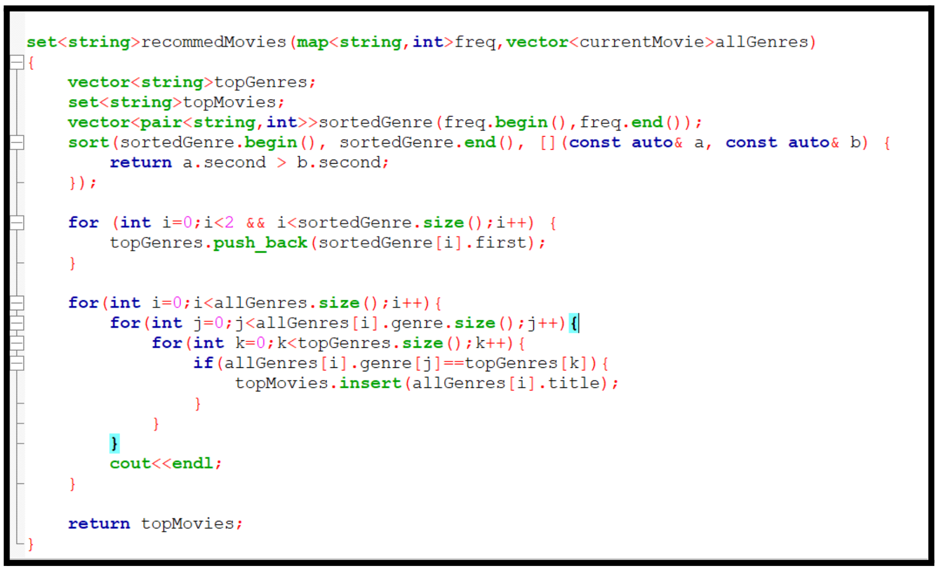
```
map<string,int> findFreq(vector<movieHistory>& movies)
{
    map<string,int>freq;
    for(int i=0;i<movies.size();i++){
        for(int j=0;j<movies[i].genre.size();j++){
            freq[movies[i].genre[j]]++;
        }
    }

    return freq;
}
```

Fig-8: Function to find frequency of genres from user history



```
vector<currentMovie> getGenres()
{
    vector<currentMovie> movies;
    ifstream details("MovieDetails.txt");
    string line;
    currentMovie temp;

    while(getline(details,line))
    {
        if(line[0]=='1'){
            temp.title=line.substr(2);
        }
        else if(line[0]=='2'){
            stringstream genres(line);
            string val;
            genres>>val;
            string snippet;
            while(getline(genres,snippet,',')){
                snippet.erase(0, snippet.find_first_not_of(" "));
                snippet.erase(snippet.find_last_not_of(" ") + 1);
                temp.genre.push_back(snippet);
            }
            movies.push_back(temp);
        }
        else if(line.size()==2 && line=="**"){
            temp.title="";
            temp.genre.clear();
        }
    }
    details.close();
    return movies;
}
```

Fig-9: Function to find genres of all available movies

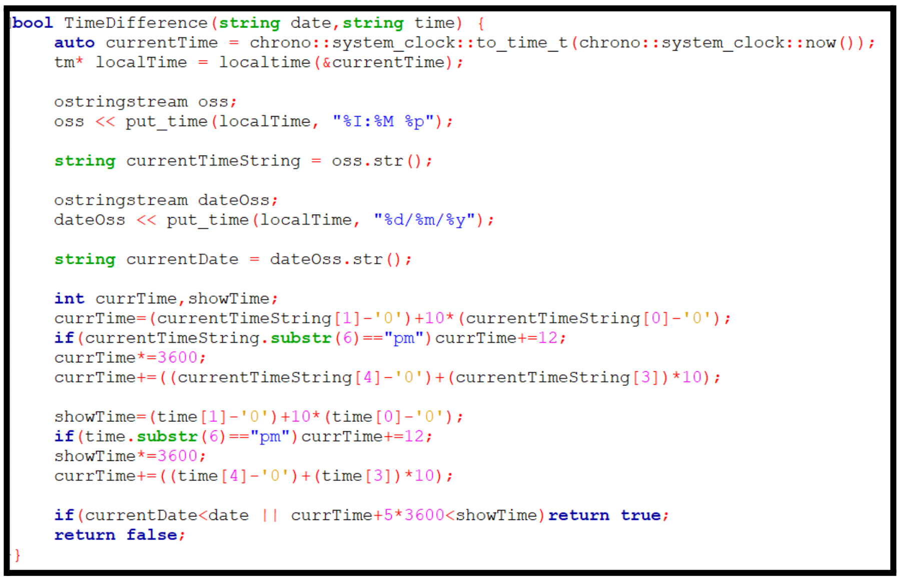


```
set<string>recommedMovies(map<string,int>freq,vector<currentMovie>allGenres)
{
    vector<string>topGenres;
    set<string>topMovies;
    vector<pair<string,int>>sortedGenre(freq.begin(),freq.end());
    sort(sortedGenre.begin(), sortedGenre.end(), [](const auto& a, const auto& b) {
        return a.second > b.second;
    });

    for (int i=0;i<2 && i<sortedGenre.size();i++) {
        topGenres.push_back(sortedGenre[i].first);
    }

    for(int i=0;i<allGenres.size();i++){
        for(int j=0;j<allGenres[i].genre.size();j++){
            for(int k=0;k<topGenres.size();k++){
                if(allGenres[i].genre[j]==topGenres[k]){
                    topMovies.insert(allGenres[i].title);
                }
            }
        }
        cout<<endl;
    }

    return topMovies;
}
```

Fig-10: Function to find movies for recommendation

- Booking cancellation:
  - ➢ For cancellation, the history of bookings of the user is read from the user file and displayed on screen, from which the user has to choose which booking they want to cancel. If the current time is at least 5 hours earlier than the showtime and is

in the booking list(which stores all bookings of available movies), then the cancellation will be successful, otherwise it's not. The booking record is removed from user history and also from the booking list. The booked seats are updated too. The function for finding time difference between showtime and time during cancellation is given below in the screenshot.

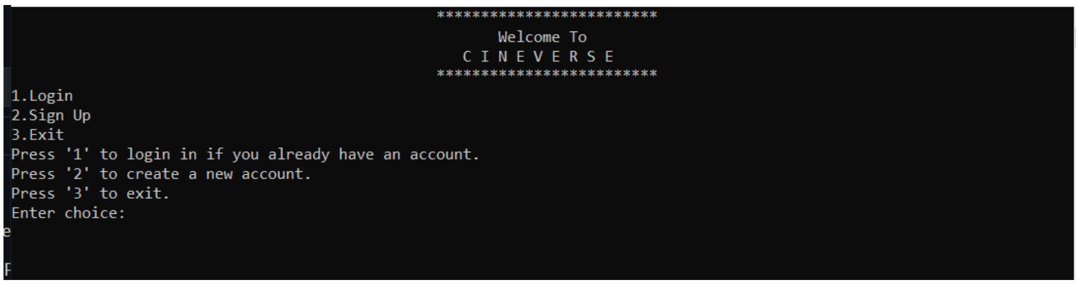


```cpp
bool TimeDifference(string date,string time) {
    auto currentTime = chrono::system_clock::to_time_t(chrono::system_clock::now());
    tm* localTime = localtime(&currentTime);

    ostringstream oss;
    oss << put_time(localTime, "%I:%M %p");

    string currentTimeString = oss.str();

    ostringstream dateOss;
    dateOss << put_time(localTime, "%d/%m/%y");

    string currentDate = dateOss.str();

    int currTime,showTime;
    currTime=(currentTimeString[1]-'0')+10*(currentTimeString[0]-'0');
    if(currentTimeString.substr(6)=="pm")currTime+=12;
    currTime*=3600;
    currTime+=((currentTimeString[4]-'0')+(currentTimeString[3])*10);

    showTime=(time[1]-'0')+10*(time[0]-'0');
    if(time.substr(6)=="pm")currTime+=12;
    showTime*=3600;
    currTime+=((time[4]-'0')+(time[3])*10);

    if(currentDate<date || currTime+5*3600<showTime)return true;
    return false;
}
```

Fig-11: Function to find the time difference

5) User Interface

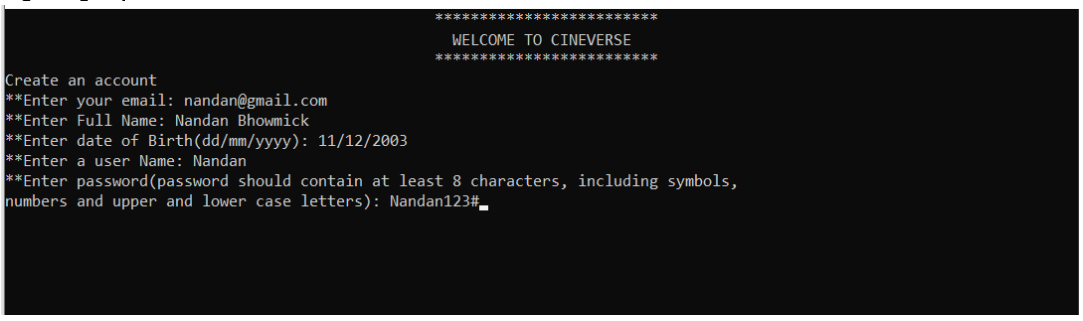
- Start window



```
                                        *************************
                                               Welcome To
                                            C I N E V E R S E
                                        *************************
1.Login
2.Sign Up
3.Exit
Press '1' to login in if you already have an account.
Press '2' to create a new account.
Press '3' to exit.
Enter choice:
```

Fig-12: Start window

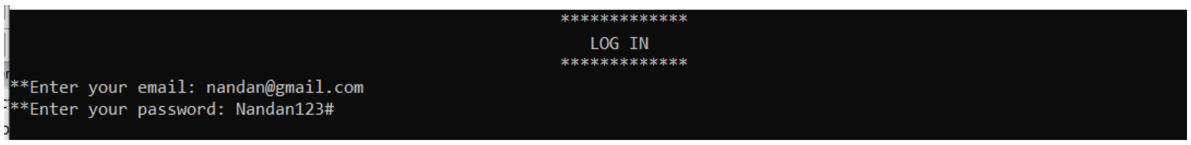
- Signing up for a new account.



```
                                        *************************
                                          WELCOME TO CINEVERSE
                                        *************************
Create an account
**Enter your email: nandan@gmail.com
**Enter Full Name: Nandan Bhowmick
**Enter date of Birth(dd/mm/yyyy): 11/12/2003
**Enter a user Name: Nandan
**Enter password(password should contain at least 8 characters, including symbols,
numbers and upper and lower case letters): Nandan123#_
```

Fig-13: Sign up window

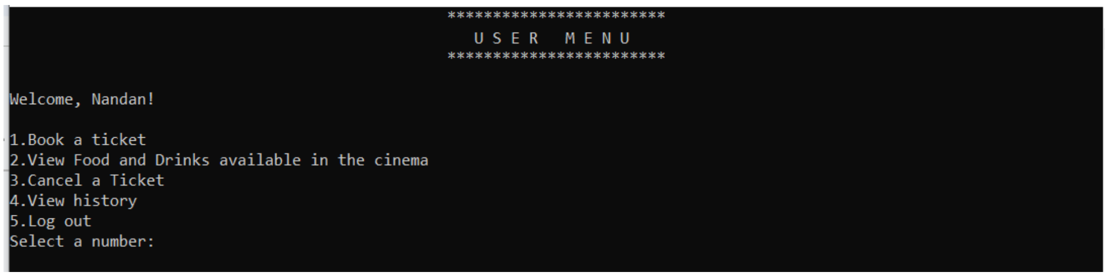
- Logging in with credentials of the new account.



```
                                        *************
                                           LOG IN
                                        *************
**Enter your email: nandan@gmail.com
**Enter your password: Nandan123#
```

Fig-14: Log in window

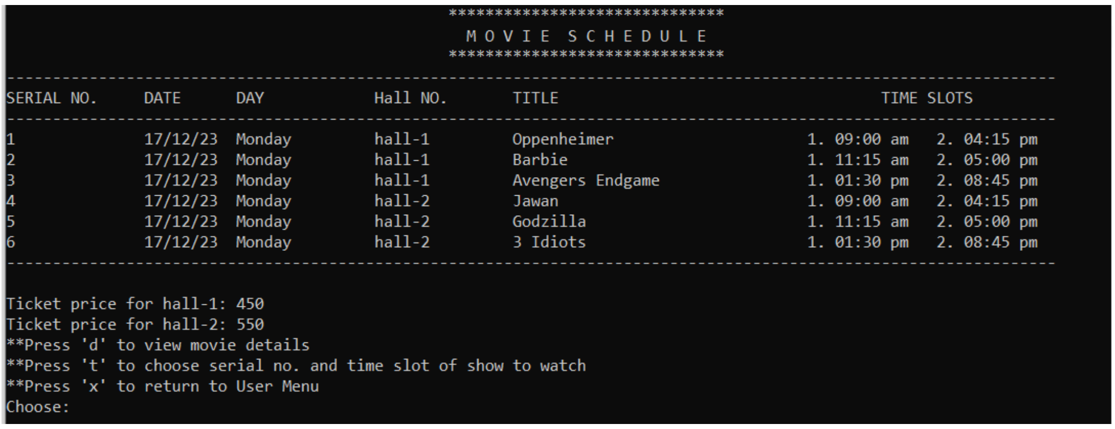
- User Menu after successful login.



```
                                 ***********************
                                    U S E R   M E N U
                                 ***********************

Welcome, Nandan!

1.Book a ticket
2.View Food and Drinks available in the cinema
3.Cancel a Ticket
4.View history
5.Log out
Select a number:
```

Fig-15: User Menu window

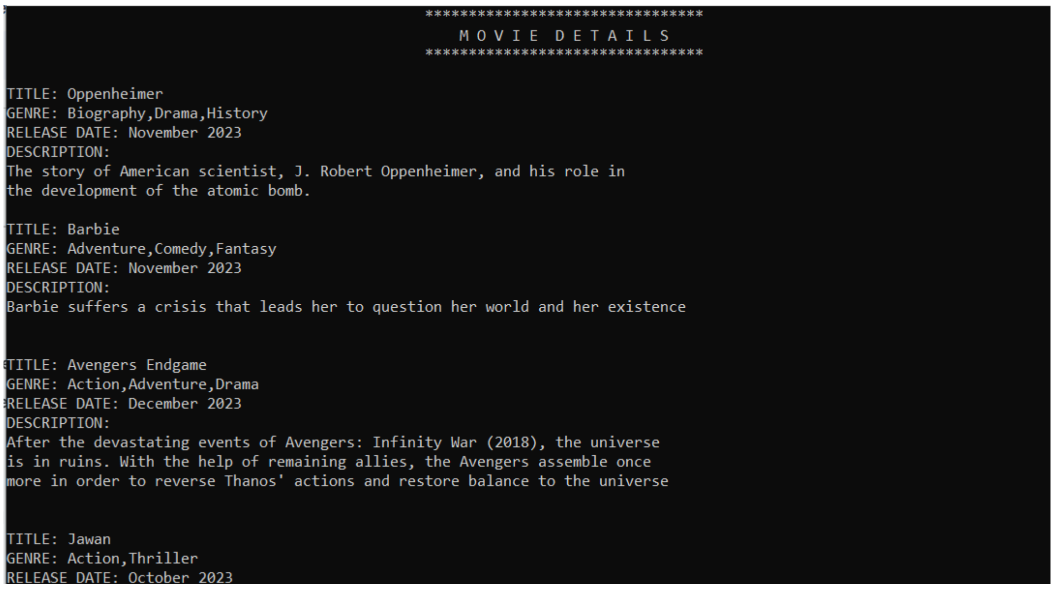
- (Booking ticket) Movie schedule



```
                                              *****************************
                                                M O V I E   S C H E D U L E
                                              *****************************
------------------------------------------------------------------------------------------------------------------
SERIAL NO.     DATE      DAY            Hall NO.       TITLE                                       TIME SLOTS
------------------------------------------------------------------------------------------------------------------
1              17/12/23  Monday         hall-1         Oppenheimer                         1. 09:00 am    2. 04:15 pm
2              17/12/23  Monday         hall-1         Barbie                              1. 11:15 am    2. 05:00 pm
3              17/12/23  Monday         hall-1         Avengers Endgame                    1. 01:30 pm    2. 08:45 pm
4              17/12/23  Monday         hall-2         Jawan                               1. 09:00 am    2. 04:15 pm
5              17/12/23  Monday         hall-2         Godzilla                            1. 11:15 am    2. 05:00 pm
6              17/12/23  Monday         hall-2         3 Idiots                            1. 01:30 pm    2. 08:45 pm
------------------------------------------------------------------------------------------------------------------

Ticket price for hall-1: 450
Ticket price for hall-2: 550
**Press 'd' to view movie details
**Press 't' to choose serial no. and time slot of show to watch
**Press 'x' to return to User Menu
Choose:
```

Fig-16: Movie schedule to choose a show time from

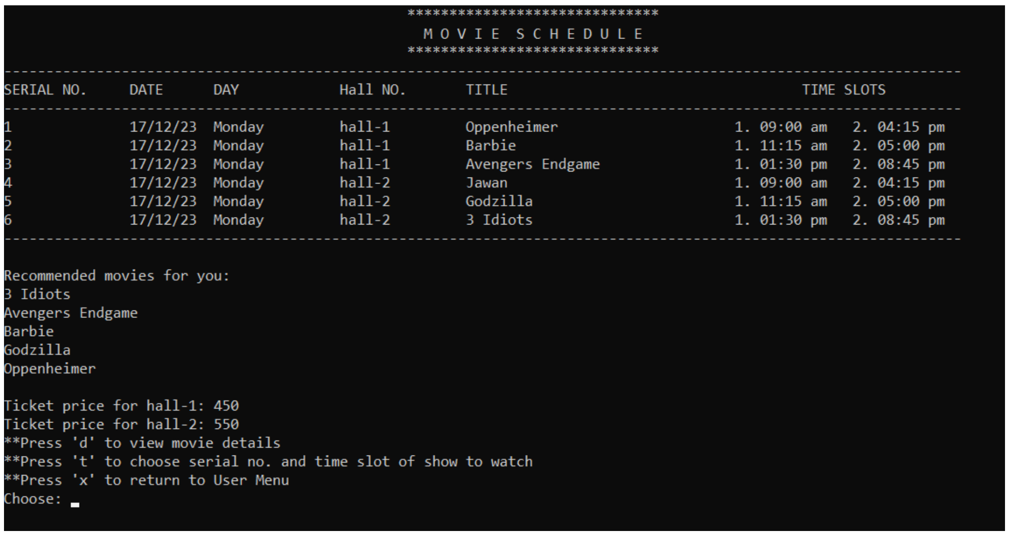
- (Booking ticket) Choosing to view movie details of available movies



```
                                                ********************************
                                                   M O V I E   D E T A I L S
                                                ********************************

TITLE: Oppenheimer
GENRE: Biography,Drama,History
RELEASE DATE: November 2023
DESCRIPTION:
The story of American scientist, J. Robert Oppenheimer, and his role in
the development of the atomic bomb.

TITLE: Barbie
GENRE: Adventure,Comedy,Fantasy
RELEASE DATE: November 2023
DESCRIPTION:
Barbie suffers a crisis that leads her to question her world and her existence


TITLE: Avengers Endgame
GENRE: Action,Adventure,Drama
RELEASE DATE: December 2023
DESCRIPTION:
After the devastating events of Avengers: Infinity War (2018), the universe
is in ruins. With the help of remaining allies, the Avengers assemble once
more in order to reverse Thanos' actions and restore balance to the universe


TITLE: Jawan
GENRE: Action,Thriller
RELEASE DATE: October 2023
```

Fig-17: Movie details

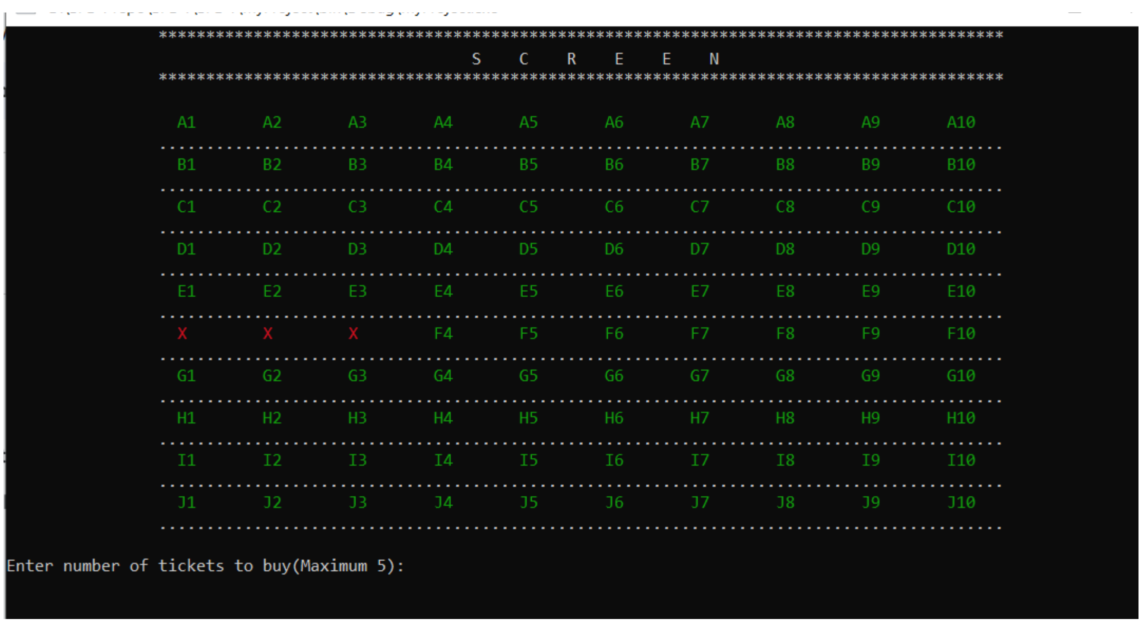
- (Booking ticket) The recommendation system recommending movies after several movies are booked.



```
                                                ******************************
                                                   M O V I E   S C H E D U L E
                                                ******************************
----------------------------------------------------------------------------------------------------------------------
SERIAL NO.     DATE      DAY            Hall NO.       TITLE                                   TIME SLOTS
----------------------------------------------------------------------------------------------------------------------
1              17/12/23  Monday         hall-1         Oppenheimer                        1. 09:00 am   2. 04:15 pm
2              17/12/23  Monday         hall-1         Barbie                             1. 11:15 am   2. 05:00 pm
3              17/12/23  Monday         hall-1         Avengers Endgame                   1. 01:30 pm   2. 08:45 pm
4              17/12/23  Monday         hall-2         Jawan                              1. 09:00 am   2. 04:15 pm
5              17/12/23  Monday         hall-2         Godzilla                           1. 11:15 am   2. 05:00 pm
6              17/12/23  Monday         hall-2         3 Idiots                           1. 01:30 pm   2. 08:45 pm
----------------------------------------------------------------------------------------------------------------------

Recommended movies for you:
3 Idiots
Avengers Endgame
Barbie
Godzilla
Oppenheimer

Ticket price for hall-1: 450
Ticket price for hall-2: 550
**Press 'd' to view movie details
**Press 't' to choose serial no. and time slot of show to watch
**Press 'x' to return to User Menu
Choose: _
```

Fig-18: Movie schedule with recommendation based on user history

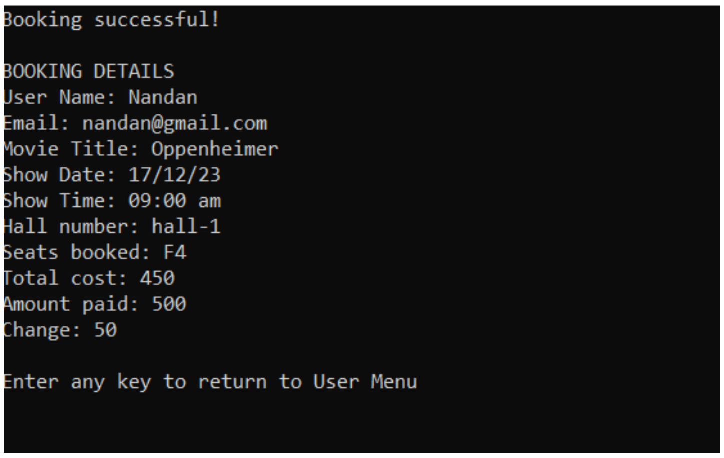
- (Booking ticket)The seat chart displaying available seats in green and unavailable seats with a red cross. Maximum 5 seats can be chosen from here.



```
                    *******************************************************************************
                                                S    C    R    E    E    N
                    *******************************************************************************

                      A1      A2      A3      A4      A5      A6      A7      A8      A9      A10
                    ...............................................................................
                      B1      B2      B3      B4      B5      B6      B7      B8      B9      B10
                    ...............................................................................
                      C1      C2      C3      C4      C5      C6      C7      C8      C9      C10
                    ...............................................................................
                      D1      D2      D3      D4      D5      D6      D7      D8      D9      D10
                    ...............................................................................
                      E1      E2      E3      E4      E5      E6      E7      E8      E9      E10
                    ...............................................................................
                      X       X       X       F4      F5      F6      F7      F8      F9      F10
                    ...............................................................................
                      G1      G2      G3      G4      G5      G6      G7      G8      G9      G10
                    ...............................................................................
                      H1      H2      H3      H4      H5      H6      H7      H8      H9      H10
                    ...............................................................................
                      I1      I2      I3      I4      I5      I6      I7      I8      I9      I10
                    ...............................................................................
                      J1      J2      J3      J4      J5      J6      J7      J8      J9      J10
                    ...............................................................................

Enter number of tickets to buy(Maximum 5):
```

Fig-19: Seating arrangement in the chosen show time

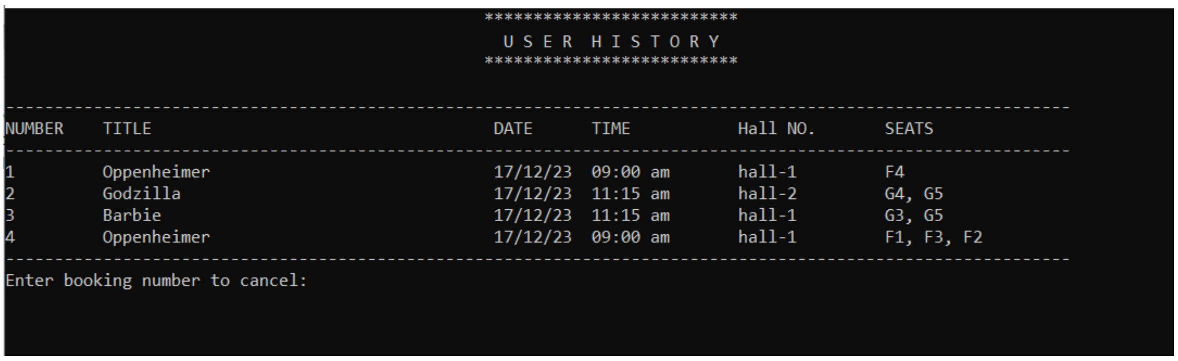
- (Booking ticket) Booking details after successful booking. The total amount has to be paid before booking can be successful.



```
Booking successful!

BOOKING DETAILS
User Name: Nandan
Email: nandan@gmail.com
Movie Title: Oppenheimer
Show Date: 17/12/23
Show Time: 09:00 am
Hall number: hall-1
Seats booked: F4
Total cost: 450
Amount paid: 500
Change: 50

Enter any key to return to User Menu
```

Fig-20: Booking receipt

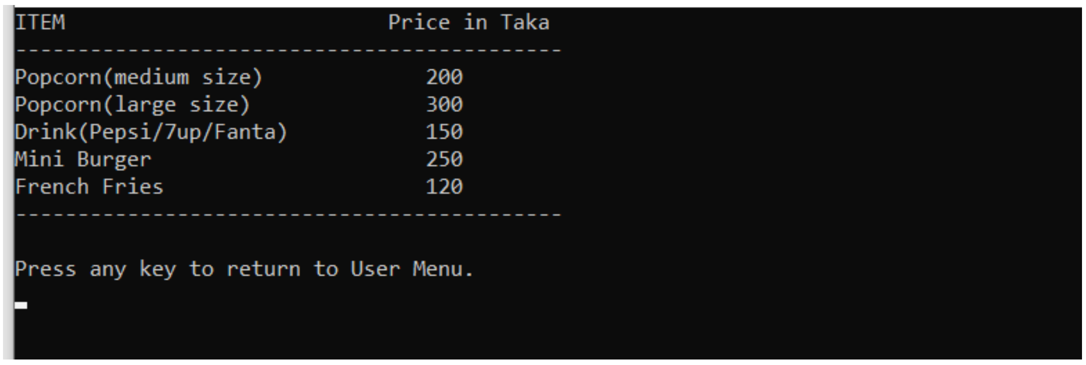
- (Canceling booking)User history to choose a booking from. If the booking is in the booking list and current time is at least 5 hours earlier than show time, booking cancellation will be successful.



```
                                                 **************************
                                                   U S E R  H I S T O R Y
                                                 **************************

-------------------------------------------------------------------------------------------------------------
NUMBER    TITLE                                             DATE      TIME           Hall NO.       SEATS
-------------------------------------------------------------------------------------------------------------
1         Oppenheimer                                       17/12/23  09:00 am       hall-1         F4
2         Godzilla                                          17/12/23  11:15 am       hall-2         G4, G5
3         Barbie                                            17/12/23  11:15 am       hall-1         G3, G5
4         Oppenheimer                                       17/12/23  09:00 am       hall-1         F1, F3, F2
-------------------------------------------------------------------------------------------------------------
Enter booking number to cancel:
```

Fig-21: Cancellation window showing user history to choose a booking

- Viewing food and drinks available in the theater.



```
ITEM                          Price in Taka
--------------------------------------------
Popcorn(medium size)             200
Popcorn(large size)              300
Drink(Pepsi/7up/Fanta)           150
Mini Burger                      250
French Fries                     120
--------------------------------------------

Press any key to return to User Menu.
```

Fig-22: Food and drinks available in theater

6) Challenges Faced

There are several challenging tasks that I had faced while working on this project. They are mentioned below, along with how I overcame them:

- Implementing SHA-256 to secure user password required integration of OpenSSL library and configuration of build settings in my IDE. I had to do some research on the Internet to understand library configuration and resolve issues related to library paths
- Organizing different information in files (e.g. user information, movie details, movie schedule, seating arrangements) was particularly important and difficult since it had to be organized in a way that would allow easy parsing and updating of information.
- Since a major part of my project involves File I/O, it was quite challenging to read and parse data, particularly because of the files having information stored in different formats. I had to do extensive research to learn how numerous file I/O operations and syntax worked and they were implemented to overcome this problem.
- Algorithms for tasks such as seat booking, ticket cancellation, and recommending movies based on user history were challenging to design. The required tasks were broken down into smaller logical steps to overcome this challenge.
- Since there are so many different features included in this project, I had to separate functions for each feature in different files to reduce complexity. Otherwise it would become tedious to look for bugs in the program.

7) Conclusion

Working on this project was a great learning experience for me. I have learned a lot about library integration in C++ projects. I learned handling file input and output, reading and parsing data from files, and managing different data formats in C++. The project challenged me to design algorithms for different features of this project, and that required me to learn so much about programming itself. I encountered numerous problems while working on this project, so tackling them led me to develop my problem solving skills too. I have got a general idea about how a software project is developed; it's challenging, arduous and exciting.

Since this is an application based project, there are plentiful features that can be added to it. The three main features that I would like to add are:

- Movie availability: Increase the number of movies available and hall number.
- Graphical User Interface: Incorporating GUI will enhance the user experience of booking a ticket. An interactive seating chart will allow the users to visually select seats during the booking process. Users will also be able to view profile information and booking history and customize preferences.
- Socket Programming: Incorporating a client-server architecture would make this project much more realistic. Users can see changes instantly in show times and seat availability without having to refresh the page, providing a seamless and dynamic experience.
- Error checking: Improve error handling in cases where users mistakenly enter incorrect data.

Reference

- https://www.geeksforgeeks.org/file-handling-c-classes/, File handling through C++ classes, 16/12/2023
- https://www.javatpoint.com/cpp-files-and-streams, C++ File and Stream, 10/12/2023
- The Complete Reference C++ by Herbert Schild, 4th edition