BlockBloom Project - Task 2

1. Setup and Implementation

1.1. Customer Database A list of 100 customers, each with:

- Unique ID (`i`)
- Name in the format `customer + i`
- Initial balance of $1000

1.2. Manager The manager has an initial balance of $0 and can:

- Add or remove movies from the database.
- Refund customers when a movie is removed.
- View the list of customers who booked tickets.
- Check the balance.

1.3. Movie Database A list of 10 movies, each with:

- Ticket price
- Available times

Note: Movie lookup could have been optimized using indices, but the current implementation uses linear search. This issue arises when new movies are added unless an upper bound is set.

2. Features

2.1. Manager Actions

1. Check balance
2. Add movies
3. Remove movies
4. See the list of customers who booked tickets

2.2. Customer Actions

1. Check balance
2. View the list of all movies
3. Book movie tickets
4. Cancel movie tickets (refunds money)
5. View all bookings

3. Design Choices

3.1. Time System Stored timings in hours for simplicity and ease of overlap checking. Conversion to day-hour format can be implemented if required.

3.2. Data Types Used `int` instead of `long long int` as inputs are small. Avoided `float` for simplicity.

3.3. Interaction The program uses a menu-driven interface with clear action numbers.

3.4. Disclaimer While familiar with C++, understanding of OOP (object-oriented programming) concepts such as public, private, and protected access was basic. Decisions were made to enhance protection by keeping sensitive parts private.

4. Test Case

The following test case demonstrates the program's functionality:

```
// test case starts here
1 123 WRONGPASS
1 123 PASS
1
4
2 1 Avengers 2 5 50 500
2 1 Avengers 500 5 50 50
5
2 2 1
2 2 2
1
5
2
3 0 50
3 0 30
3 1 1
5
4 2 50
4 0 10
5
1
6
```

```
1 123 PASS
1
4
3 0
1
5
3
//test case ends here
```

5. Code Implementation

The full code is as follows:

```
//code starts here

#include <bits/stdc++.h>
using namespace std;
class person{
    string password;
    string name;
    public:
    int id;
    int money;
    person(int id1,string name1,int money1,string password1){
        id=id1;
        money=money1;
        name=name1;
        password=password1;
    }
    virtual bool canAcess(int id1,string pass1){
        if(id1==id && pass1==password){
            cout<<"\nWelcome "<<name<<endl;
            return true;
        }
        cout<<"\nSorry! Wrong ID or Password!!"<<endl;
        return false;
    }
};

class manager:public person{
    public:
    manager(int id1,string name1,int money1,string password1):
       person(id1,name1,money1,password1){}

};

class customer:public person{
    public:
    unordered_map <int,int> bookings;
    customer(int id1,string name1,int money1,string password1):
       person(id1,name1,money1,password1){}
```

```
};

class movie{
    public:
    int id;
    string title;
    int duration;
    int start;
    int capacity;
    int booked;
    int ticket_price;
    unordered_map <int,int> customers;
    movie(int id1,string title1,int start1,int duration1,int
       capacity1,int booked1,int ticket_price1){
        id=id1;
        title=title1;
        start=start1;
        duration=duration1;
        capacity=capacity1;
        booked=booked1;
        ticket_price=ticket_price1;
    }
    void moviedes(){
        cout<<"The Id of "<<title<<" is "<<id<<". It starts at "
           <<start<<" and is "<<duration<<" hrs long. The Ticket
           price for this movie is "<<ticket_price<<" and we have
            "<<capacity-booked<<" tickets left."<<endl;
    }
};
class movie_database{
    public:
    vector<movie> movies;
    bool overlap(const movie&movie1,const movie&movie2){
        if((movie1.start>=movie2.start && movie1.start<=movie2.
           start+movie2.duration) || (movie2.start>=movie1.start
           && movie2.start<=movie1.start+movie1.duration)) return
            true;
        else return false;
    }
    void addmovie(int id1,string title1,int start1,int duration1,
       int capacity1,int ticket_price1){
        movie movie1(id1,title1,start1,duration1,capacity1,0,
           ticket_price1);
        for(const auto m:movies){
            if(overlap(movie1,m)){
                cout<<"\nSorry, The given movie cannot be added
                   as it overlaps with the movie ID: "<<m.id<<
                   endl;
                return;
            }
            if(movie1.id==m.id){
```

```
77                  cout<<"\nSorry, a movie with ID "<<m.id<<"
                        already exists"<<endl;
78                  return;
79              }
80          }
81          cout<<"\nThe movie with id "<<movie1.id<<" has been
                sucessfully added"<<endl;
82          movies.push_back(movie1);
83      }
84      void removemovie(vector<customer> &customer_database,manager&
            manager1){
85          int id1;
86          cout<<"\nPlease enter the ID of the movie you want to
                remove: "<<endl;
87          cin>>id1;
88          for (int i = 0; i < movies.size(); i++){
89              if(movies[i].id==id1){
90                  for (auto& m:movies[i].customers){
91                      customer_database[m.first].money+=movies[i].
                            ticket_price*m.second;
92                      manager1.money-=movies[i].ticket_price*m.
                            second;
93                      customer_database[m.first].bookings.erase(id1
                            );
94                  }
95
96                  movies.erase(movies.begin()+i);
97                  cout<<"\nThe movie with ID "<<id1<<" has
                        successfully been removed"<<endl;
98                  return;
99              }
100         }
101         cout<<"\nSorry, no movie with the ID "<<id1<<" exixts"<<
                endl;
102     }
103
104 };
105 int main(){
106     movie_database movie_database1;
107     manager manager1 = manager(123,"MANAS",0,"PASS");
108     vector<customer> customer_database;
109     for (int i = 0; i < 100; i++){
110         customer_database.push_back(customer(i,"customer"+
                to_string(i),1000,to_string(i)));
111     }
112     for (int i = 0; i < 10; i++)
113     {
114         movie_database1.addmovie(i,"movie"+to_string(i),max(i*i
                ,1),i+5,50,max(i*i*3,30));
115     }
116
```

```
117     while(1){
118         cout<<"\nAre you a manager or a customer?\n1 for manager\
              n2 for customer\n3 to exit the application"<<endl;
119         int who;
120         cin>>who;
121         if(who==3) break;
122         if(who!=1 && who!=2){
123             cout<<"Wrong Input!!!"<<endl;
124             continue;
125         }
126         int id;
127         string pass;
128         cout<<"\nID :"<<endl;
129         cin>>id;
130         cout<<"Password :"<<endl;
131         cin>>pass;
132 
133         if(who==1 && manager1.canAcess(id,pass)){
134             while(1){
135                 cout<<"\nWhat would you like to do?\n1 to check
                      balance\n2 to add a movie\n3 to remove a movie
                      \n4 to see the list of customers who booked
                      the tickets for the movies\n5 to go back to
                      the previous menu"<<endl;
136                 int x;
137                 cin>>x;
138                 if(x==1){
139                     cout<<"\nYou have "<<manager1.money<<"$"<<
                          endl;
140                     continue;
141                 }
142                 else if(x==2){
143                     int id1;
144                     string title;
145                     int duration;
146                     int start;
147                     int capacity;
148                     int booked;
149                     int ticket_price;
150                     cout<<"\nID: "<<endl;
151                     cin>>id1;
152                     cout<<"Title: "<<endl;
153                     cin>>title;
154                     cout<<"Start: "<<endl;
155                     cin>>start;
156                     cout<<"Duration: "<<endl;
157                     cin>>duration;
158                     cout<<"Capacity: "<<endl;
159                     cin>>capacity;
160                     cout<<"Ticket Price: "<<endl;
161                     cin>>ticket_price;
```

```
                    if(duration==0){
                        cout<<"\nCannot add a movie with duration
                            of 0 hrs"<<endl;
                        continue;
                    }
                    else if(capacity==0){
                        cout<<"\nCannot add a movie with capacity
                            of 0 customers"<<endl;
                        continue;
                    }
                    else if(ticket_price==0){
                        cout<<"\nCannot add a movie with ticket
                           price of 0$"<<endl;
                        continue;
                    }
                    movie_database1.addmovie(id1,title,start,
                       duration,capacity,ticket_price);
                }
                else if(x==3){
                    movie_database1.removemovie(customer_database
                       ,manager1);
                }
                else if(x==4){
                    for(auto& movies:movie_database1.movies){
                        cout<<"\nThe following booking have been
                           made for movie id "<<movies.id<<endl;
                        for (auto& m:movies.customers )
                        {
                            cout<<m.second<<" tickets were bought
                                by customer with id "<<m.first<<
                               endl;
                        }
                        if(movies.customers.empty()) cout<<"No
                           booking"<<endl;

                    }
                }
                else if(x==5) break;
                else{
                    cout<<"Invalid input"<<endl;
                }
            }
        }
        else if(id>100 && who==2){
            cout<<"\nWrong ID"<<endl;
            continue;
        }
        else if(who==2 && customer_database[id].canAcess(id,pass)
           ){
            while(1){
                cout<<"\nWhat would you like to do?\n1 to check
```

```
                balance\n2 to get all the list of movie\n3 to
                book a ticket\n4 to cancel a movie booking\n5
                to see all your bookings\n6 to go back to the
                previous menu"<<endl;
203             int x;
204             cin>>x;
205             if(x==1){
206                 cout<<"\nYou have "<<customer_database[id].
                        money<<"$"<<endl;
207                 continue;
208             }
209             else if(x==2){
210                 cout<<"\nAll the movies are listed below:"<<
                        endl;
211                 for(auto& m:movie_database1.movies){
212                     m.moviedes();
213                 }
214             }
215             else if(x==3){
216                 int movieid,num;
217                 cout<<"\nEnter the movie id and number of
                        tickets you want to book:"<<endl;
218                 cin>>movieid>>num;
219                 bool flag=true;
220                 for(auto& m:movie_database1.movies){
221                     if(m.id==movieid){
222                         flag=false;
223                         cout<<"\n";
224                         if(m.ticket_price*num>
                                customer_database[id].money){
225                             cout<<"You don't have enough
                                    money"<<endl;
226                             break;
227                         }
228                         else if(m.capacity-m.booked<num){
229                             cout<<"Sorry not enough seats
                                    left"<<endl;
230                             break;
231                         }
232                         else{
233                             cout<<"Sucessfully Booked"<<endl;
234                             m.booked+=num;
235                             customer_database[id].money-=m.
                                    ticket_price*num;
236                             manager1.money+=m.ticket_price*
                                    num;
237                             customer_database[id].bookings[m.
                                    id]+=num;
238                             m.customers[id]+=num;
239                             break;
240                         }
```

```
241                         }
242                     }
243                     if(flag){
244                         cout<<"\nWe do not have a movie with the
                                given ID"<<endl;
245                     }
246                 }
247                 else if(x==4){
248                     int movieid,num;
249                     cout<<"\nEnter the movie id and number of
                            tickets you want to cancel: "<<endl;
250                     cin>>movieid>>num;
251                     bool flag=true;
252                     for(auto& m:customer_database[id].bookings){
253                         if(m.first==movieid){
254                             flag=false;
255                             if(m.second<num){
256                                 cout<<"\nYou have only booked "<<
                                        m.second<<" number of tickets.
                                         So we cannot cancel "<<num<<"
                                         number of tickets for you"<<
                                        endl;
257                             }
258                             else{
259                                 m.second-=num;
260                                 for(auto& n:movie_database1.
                                        movies){
261                                     if(n.id==movieid){
262                                         n.booked-=num;
263                                         n.customers[id]-=num;
264                                         manager1.money-=n.
                                            ticket_price*num;
265                                         customer_database[id].
                                            money+=n.ticket_price*
                                            num;
266                                         if(n.customers[id]==0){
267                                             n.customers.erase(id)
                                                ;
268                                         }
269                                         break;
270                                     }
271                                 }
272                                 if(m.second==0){
273                                     customer_database[id].
                                        bookings.erase(m.first);
274                                 }
275                                 cout<<endl<<num<<" tickets have
                                    sucessfully been cancelled for
                                     the movie with id "<<movieid
                                    <<endl;
276                                 break;
```

```
277                                 }
278                             }
279                         }
280                         if(flag){
281                             cout<<"\nYou did not book a movie with
                                the given ID"<<endl;
282                         }
283                         continue;
284                     }
285                     else if(x==5){
286                         cout<<"\nYou have booked:"<<endl;
287                         for(const auto m:customer_database[id].
                            bookings){
288                             cout<<m.second<<" tickets for the movie
                                with id "<<m.first<<endl;
289                         }
290                         if(customer_database[id].bookings.empty())
                            cout<<"No bookings"<<endl;
291                     }
292                     else if(x==6) break;
293                     else{
294                         cout<<"\nInvalid input"<<endl;
295                     }
296                 }
297 
298             }
299         }
300 
301 }
302 
303 //code ends here
```