Cairo University

Faculty of Computers and Information

CS251

Software Engineering I

GOFO

ID: PM-414

Software Design Specifications

And Implementation

Version 2.0

Team Names and Emails

CS251: Phase 2 – <Team Name>
Project: <GOFO>

Software Design Specification

Month & Year

Contents

CS251: Phase 2 – <Team Name>
Project: <GOFO>

Software Design Specification

1. Team

ID	Name	Email	Mobile

2. Document Purpose and Audience

This document is the SDS for the FOGO project

3. System Models

I. Class Diagram(s)

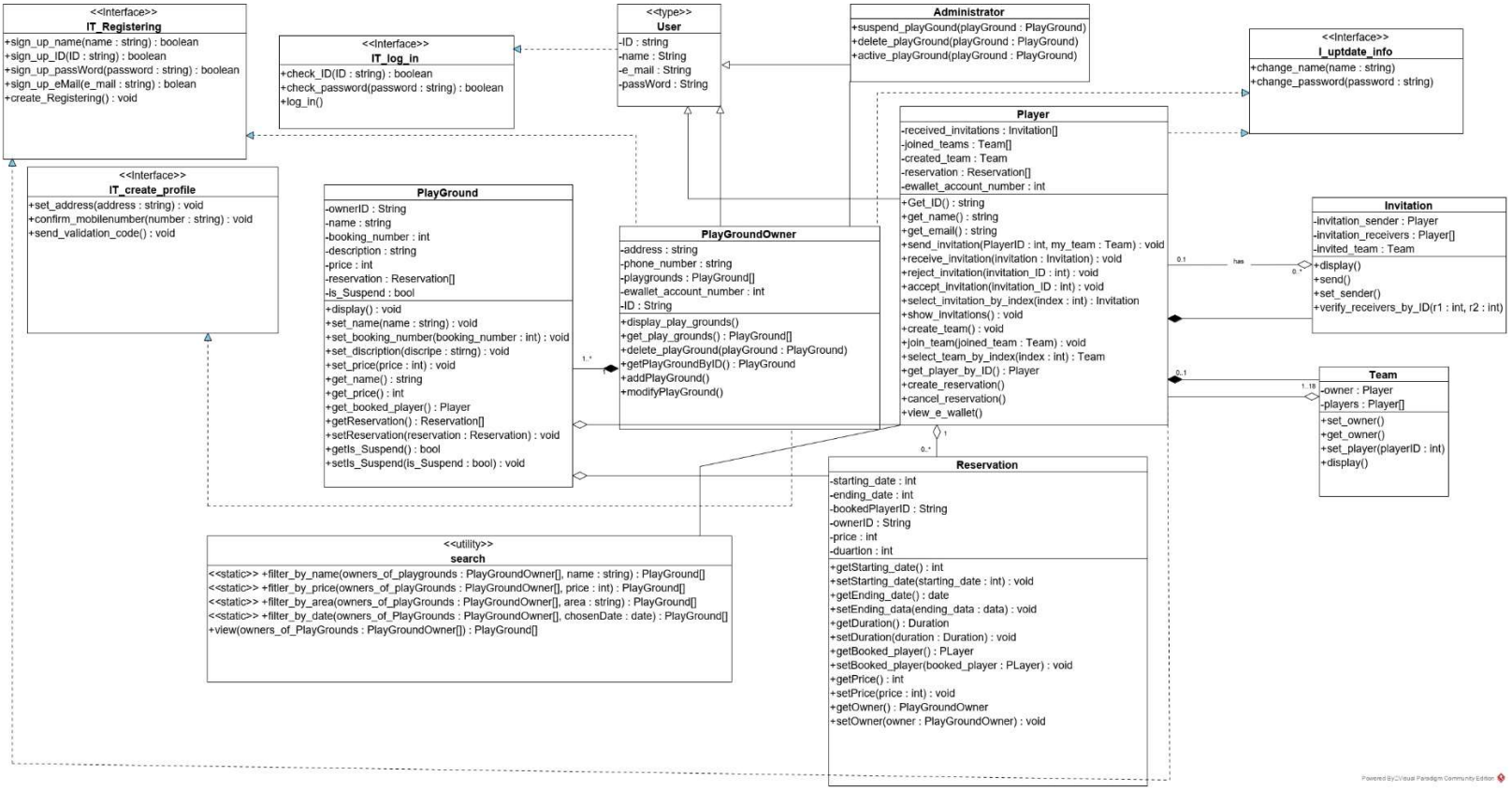
- this class diagram was generated using visual Paradigm 16.1

- this is V1.0 that was made before the code implementation

- To download the Class Diagram in full quality
(https://drive.google.com/file/d/1nsGyp6gHVWQ33gfE59KtCbgNG_KdouCy/view?usp=sharing)

CS251: Phase 2 – <Team Name>
Project: <GOFO>

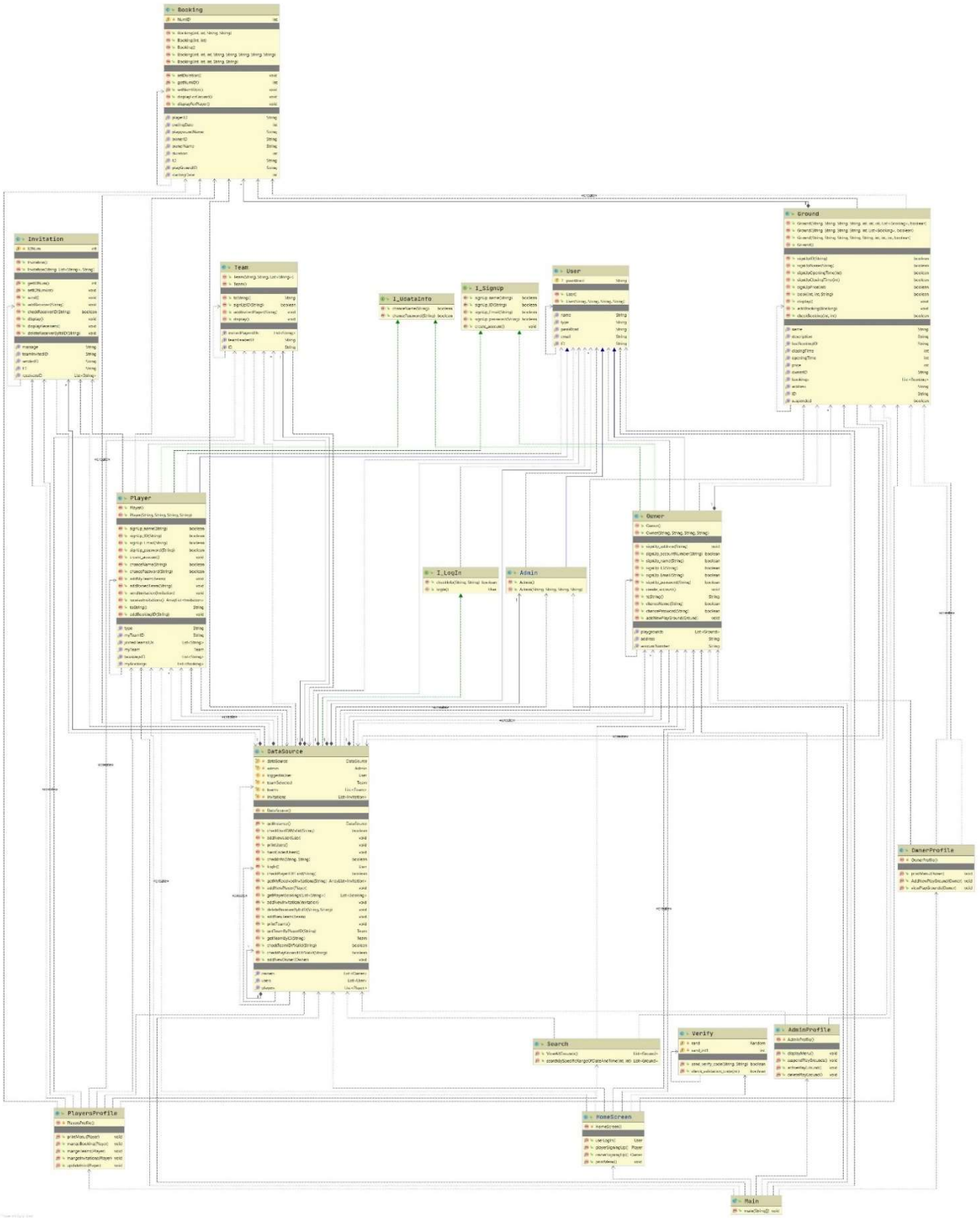
Software Design Specification



- this is V2.0
- Was auto generated using intelliJ IDEA Ultimate 2020.1.1 UML tool
- This version after the was made after the Coding with all the changes made
- This class diagram was made for a console-based APP
- All the control classes are just a Utility Classes has all the static method needed to be using in main because it is a console based APP
- To download the Class Diagram in full quality (https://drive.google.com/file/d/1TJop90PTMxLWqs4yfZ-weQahBcEklT9i/view?usp=sharing)

CS251: Phase 2 – <Team Name>
Project: <GOFO>

Software Design Specification



CU – FC
Prepare
Edited b

Software Design Specification

II. Class Descriptions

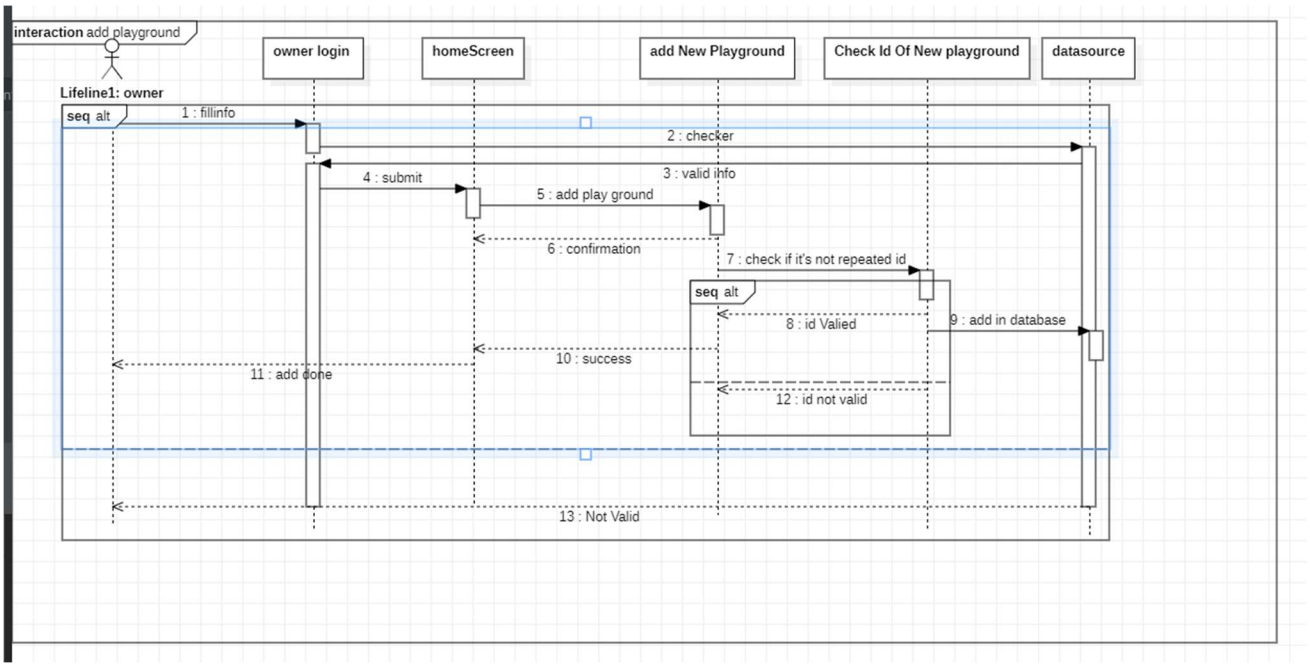
Class ID	Class Name	Description & Responsibility
C1	AdminProfile	this class is a control class and because this program is a console based program so it just contain a static methods that will be used in the main to keep the code more organized and will contain all the methods the the Admin will be able to preform .
C2	PlayerProfile	this class is a control class and because this program is a console based program so it just contain a static methods that will be used in the main to keep the code more organized and will contain all the methods that the Player will be able to preform .
C3	OwnerProfile	* this class is a control class and because this program is a console based program so it just contain a static methods that will be used in the main to keep the code more organized and will contain all the methods that the owner will be able to preform in his profile page .
C4	HomeScreen	this class is a utility class ans also we can consider it as a control class that allow the user to sign up and log in through static methods that will be used in the main to reduce the code there and keep it more arranged
C5	DataSource	class is singleton class which means it has only one object this class is responsible for storing all the data about the player or the play ground owner and the admin it will all the methods needed to get the data needed for the application there will be in there some hard coded player and playGrounds for testing this data will not be connect to any databases maybe it will be stored in text file but that is not for sure yet
C6	Booking	this class contain all the info needed about the booking and it only exist int he a list of every play ground
C7	Ground	this class has all the info needed about any playground and it is only exist in the Owner class
C8	Invitation	it has all the info needed about the invitation and responsible for sending and receiving the invitations
C9	Search	*Utility Class that has all the method needed to preform all kind of searching*
C10	Team	has all the info needed for the team
IT1	I_LogIn	interface contain all the method needed for for Logging in
IT2	I_SignUp	interface contain all the method needed for for signing up

Software Design Specification

Class ID	Class Name	Description & Responsibility
IT3	I_UdataInfo	interface contain all the method needed for Updating info
C11	Admin	class that is representing the admin
C12	Owner	class that contain every thing about the owner and his playerGrounds
C13	Player	class that contain every thing about the Player and his bookings
C14	User	this is the base class for the users that the basic info
C15	Verify	sent the verification code and check if it is right or not and has connection to the internet and to run this class you must add the lib added with the project

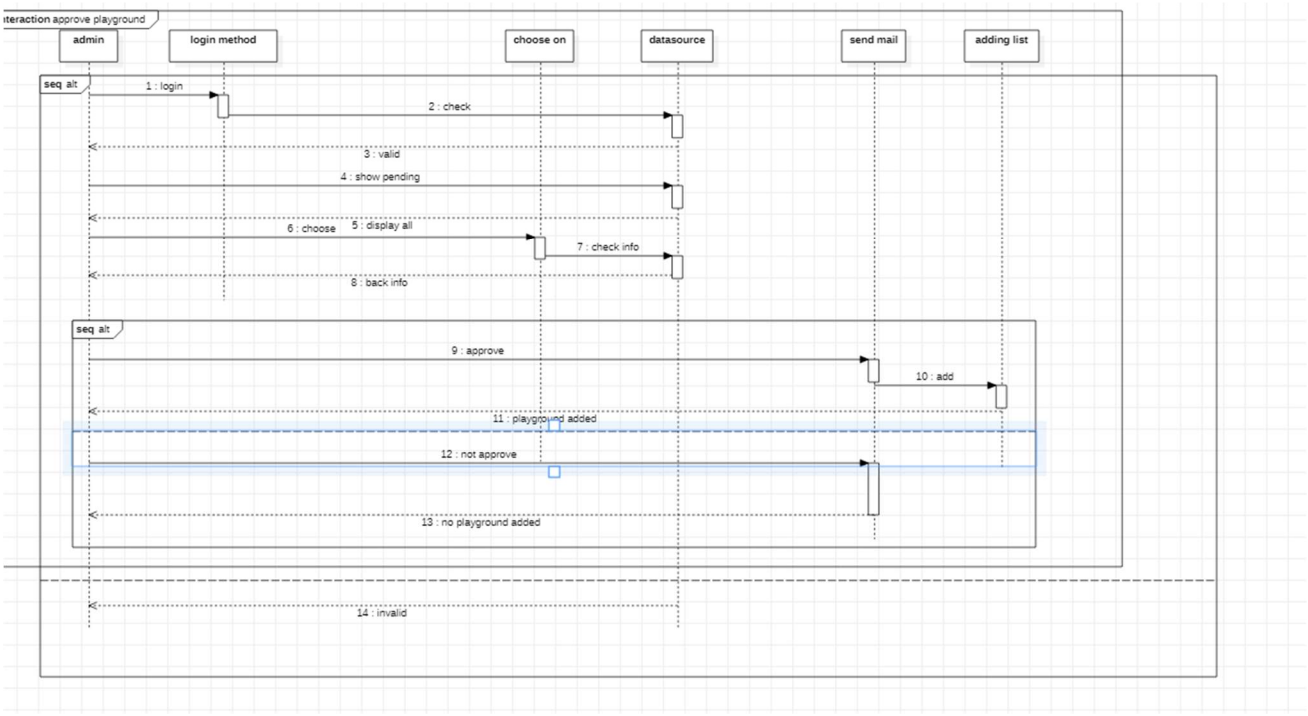
III. Sequence diagrams

1) add playground

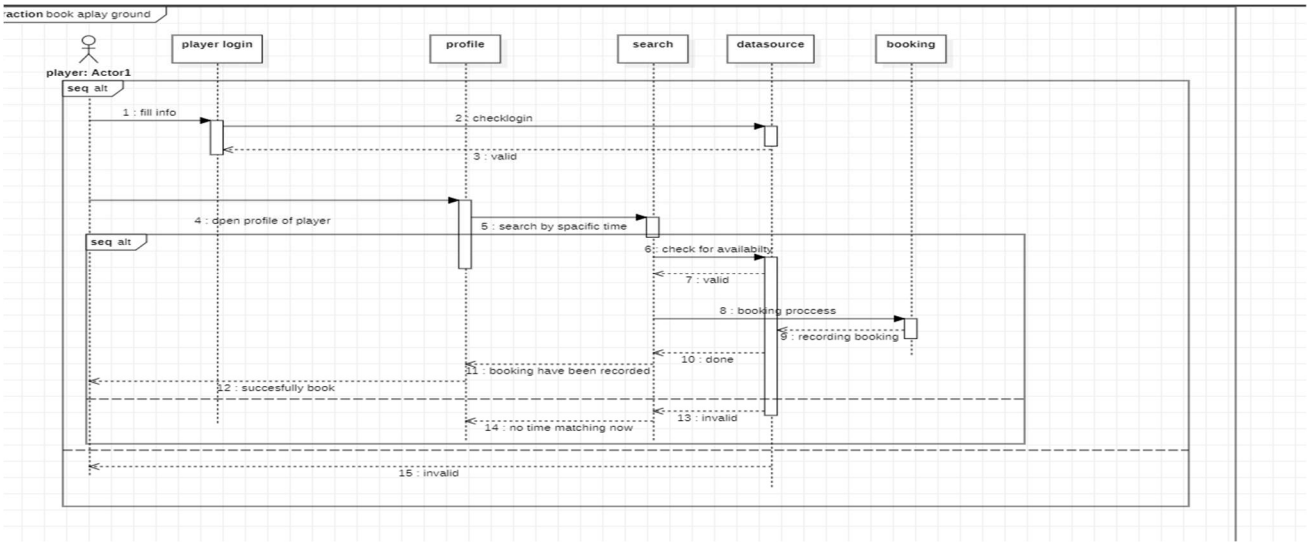


2) approval

Software Design Specification

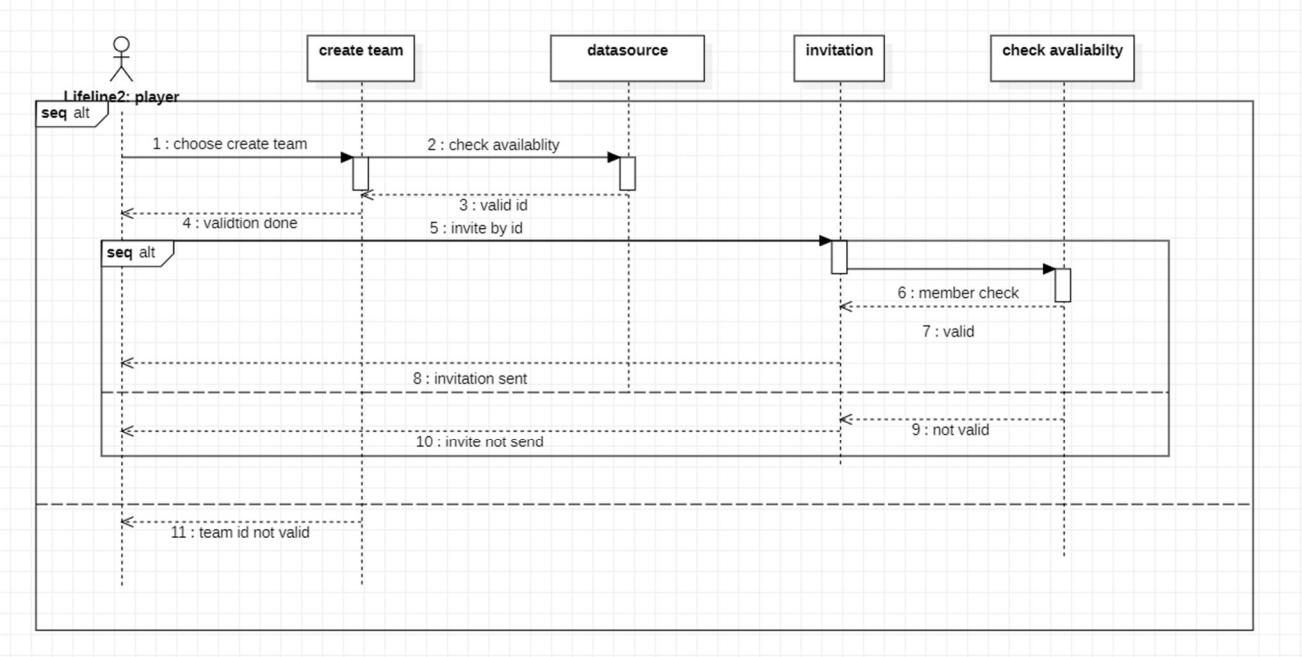


3) booking



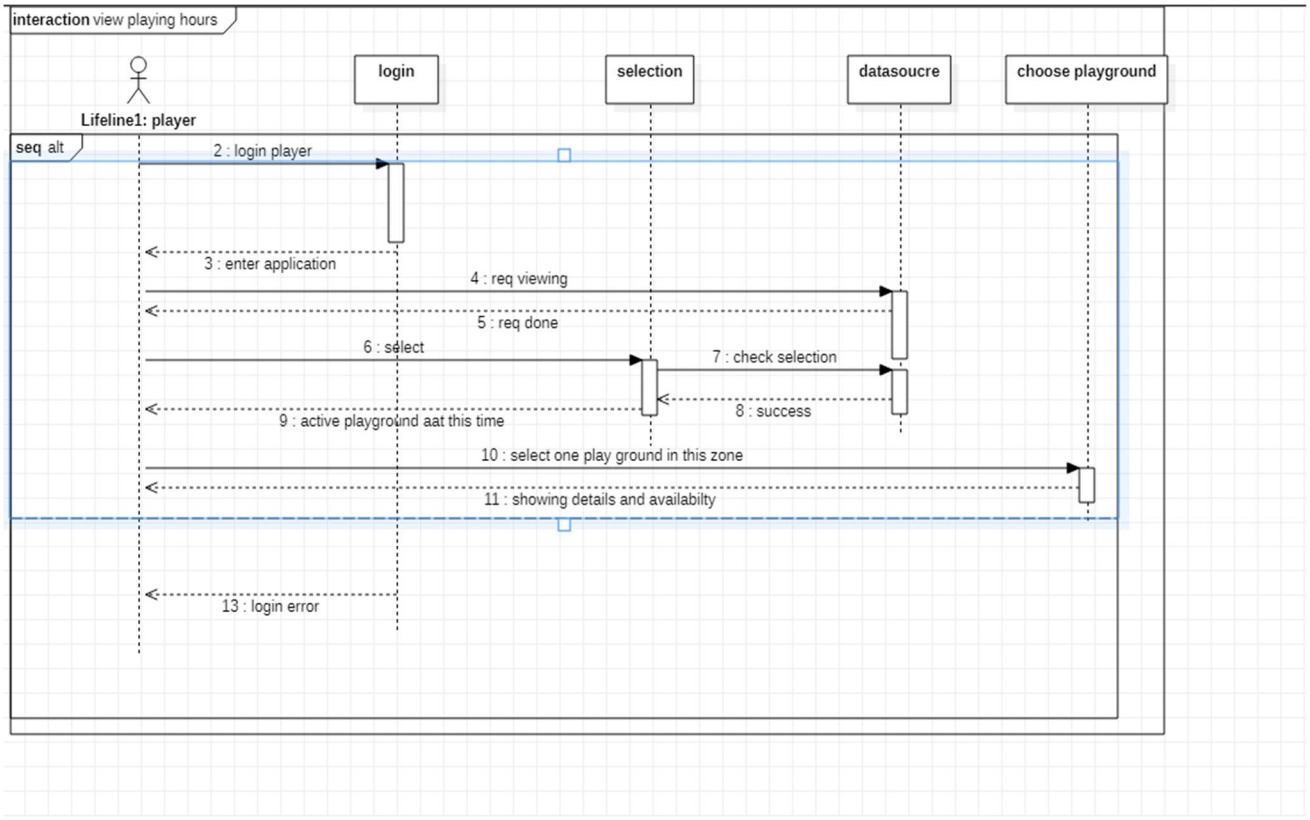
Software Design Specification

4) create team



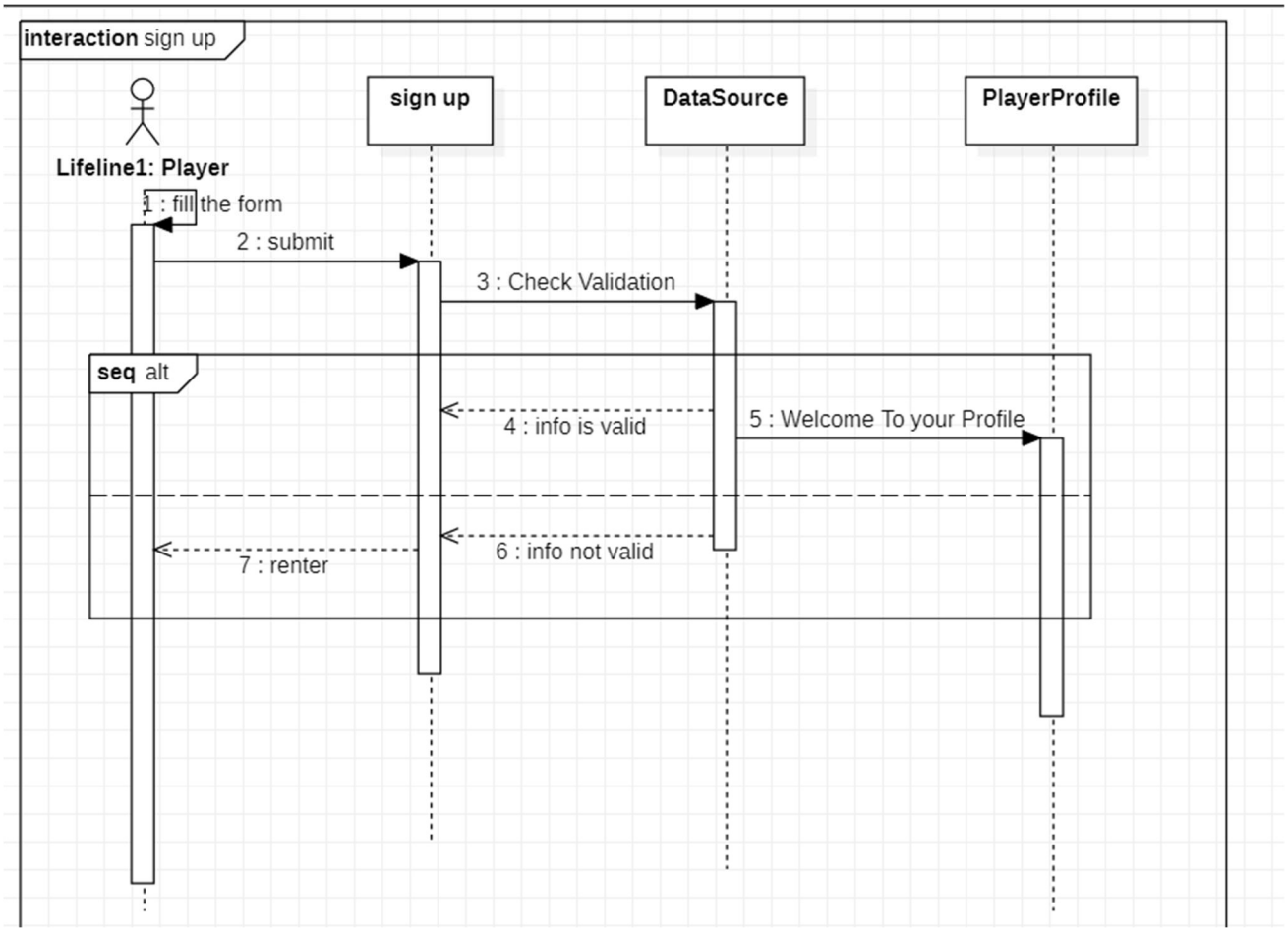
5) playing hours

Software Design Specification



6) sign up

Software Design Specification



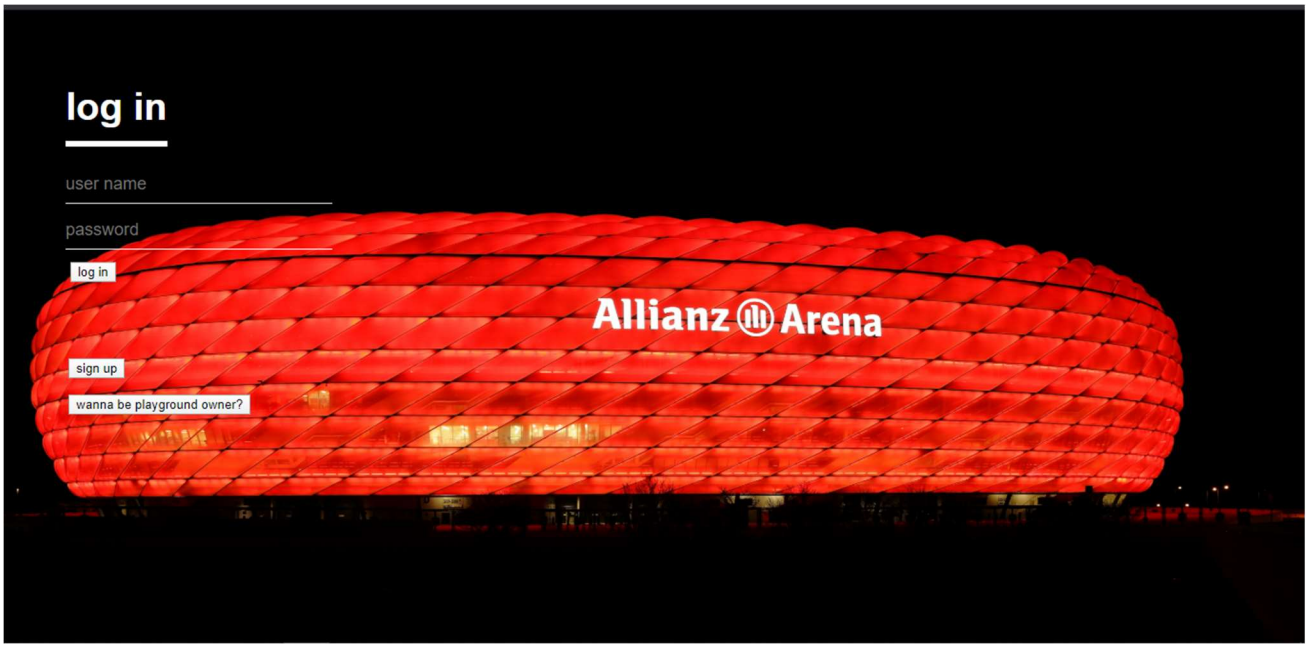
Class - Sequence Usage Table

Sequence Diagram	Classes Used	All Methods Used
1. adding playground	Owner	Login , checkIDIfValid , addNewPlayground
2. booking	Player	Login , checkIDIfValid , search , book , checkBookingIDIfValid
3. playing hours	player	login ,viewing list of playgrounds select

Software Design Specification

Sequence Diagram	Classes Used	All Methods Used
4. sign up	player	fill form ,check for validation , profile display
5. approval	admin	login ,check ,dispaly pending request display one , send approval mail
6. create team	player	creat teami nvitaion ,check availabilty id playground ,check availabile id team member , sendmail

IV. User Interface Design



(sc1)

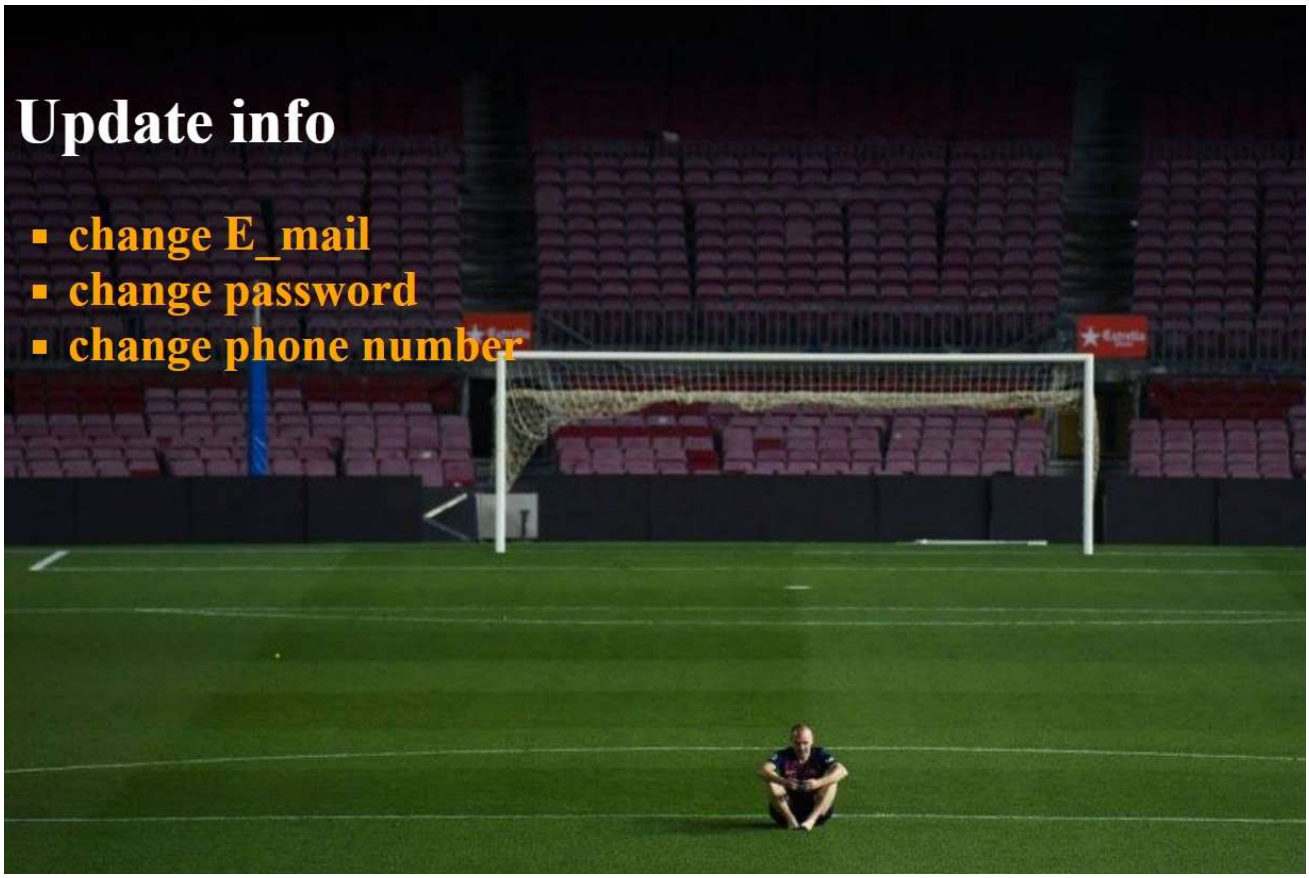
Software Design Specification

(sc2)

Software Design Specification

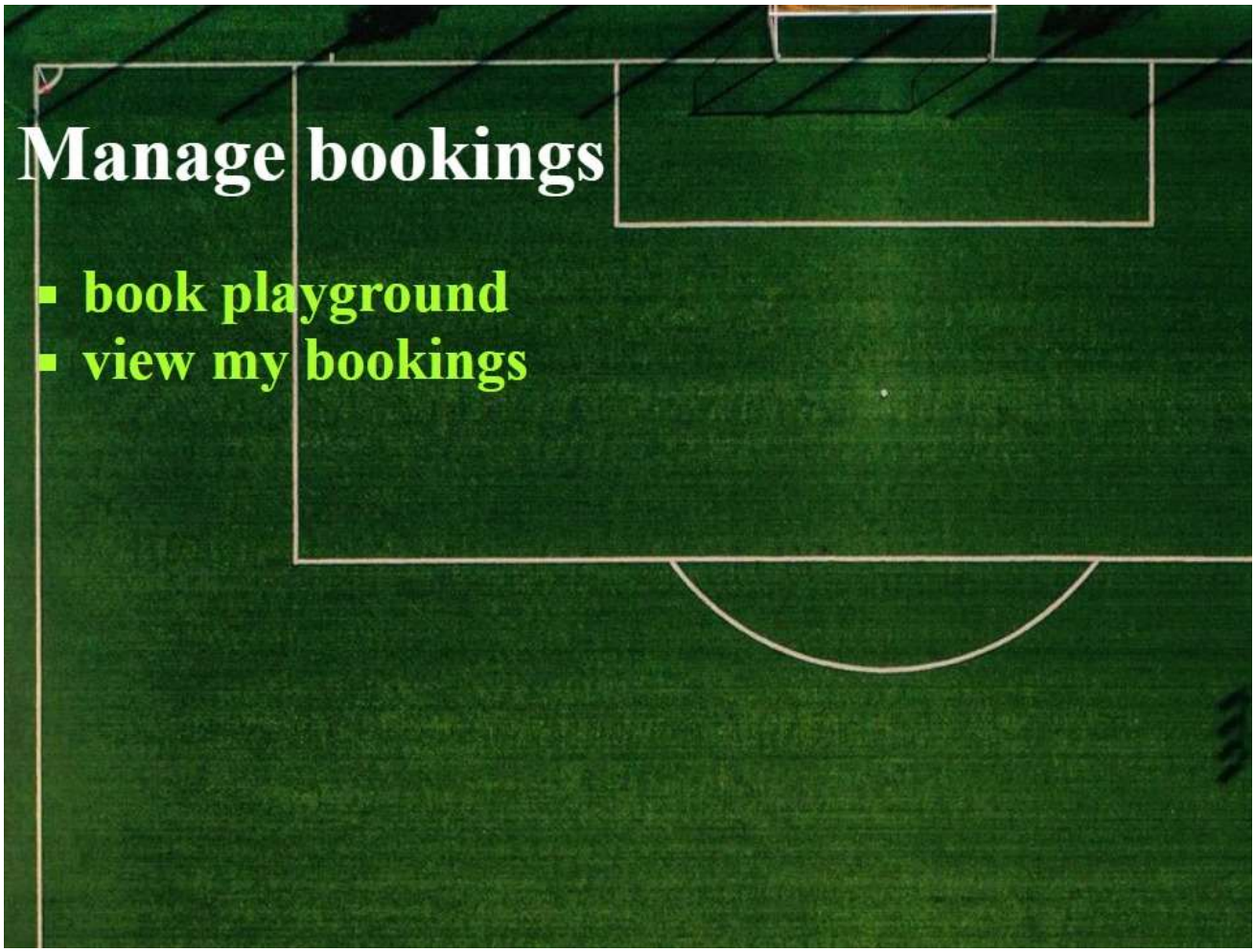
(sc3)

Software Design Specification



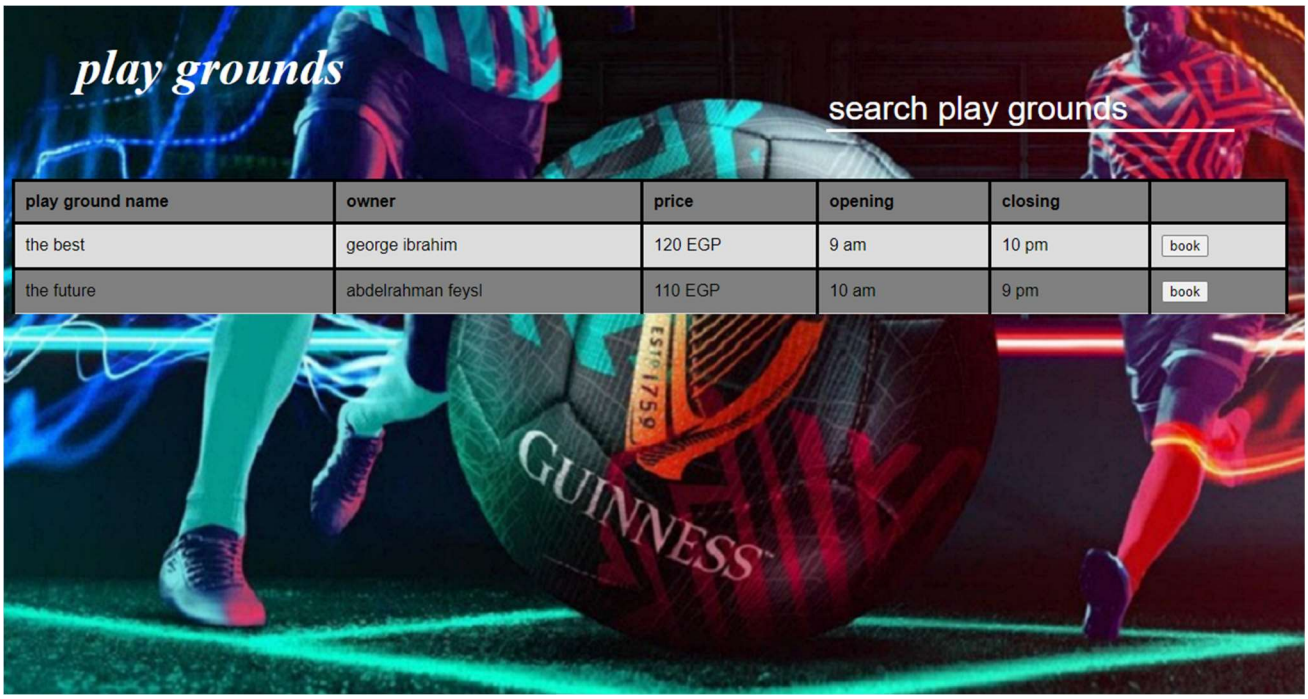
(sc4)

Software Design Specification



(sc5)

Software Design Specification



(sc6)

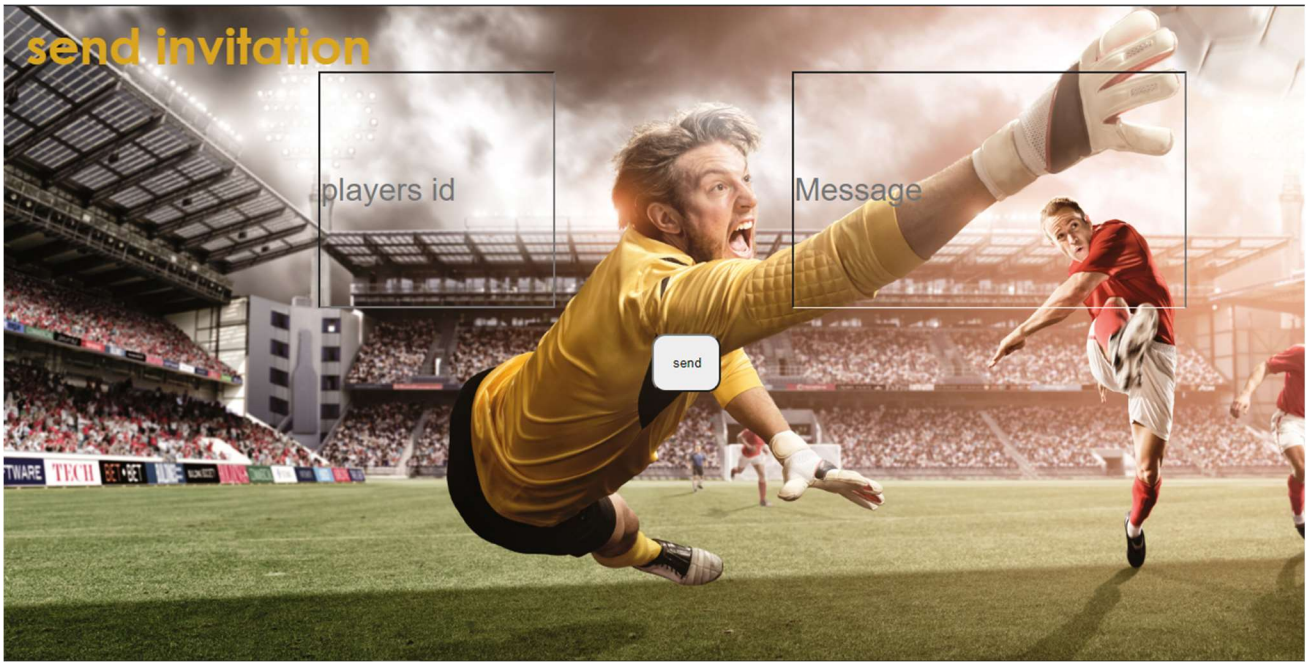
Software Design Specification

(sc7)

Software Design Specification

(sc8)

Software Design Specification



(sc9)

Software Design Specification

(sc10)

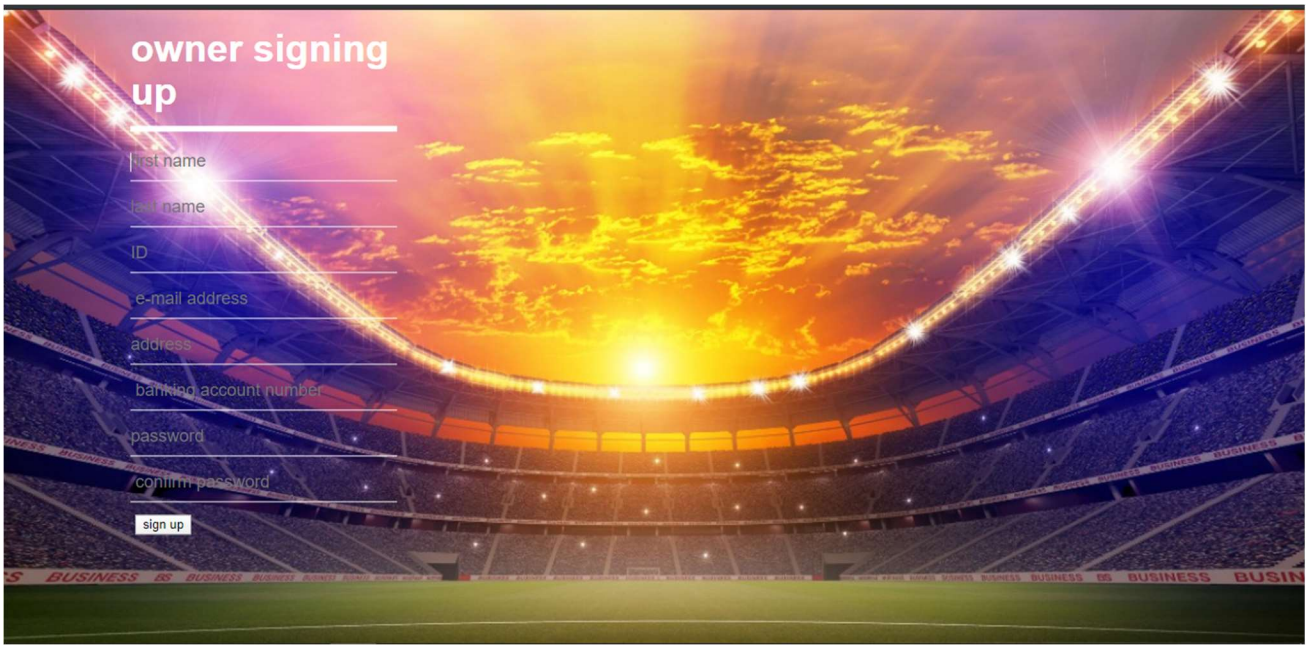
Software Design Specification

(sc11)

Software Design Specification

(sc12)

Software Design Specification



(sc13)

Software Design Specification

(sc14)

Software Design Specification

(sc15)

Software Design Specification

(sc16)

Software Design Specification

(sc17)

Screen ID	Screen Name	Screen / Wireframe Description
SC1	LOG IN	In this page we log in or we sign up as user or owner
SC2	user Sign up	In this page we sign up as user
SC3	User home	The first page that the user opens after logging in
SC4	Update info	In this page the user can edit information of his account

Software Design Specification

SC5	Mange booking	This page shows booking settings
SC6	playgrounds	This page displays the PlayGrounds available for reservation How much does it cost?When it open and when it close?
SC7	My booking	This page displays the PlayGrounds available for reservation How much does it cost? When it open and when it close? And the date of booking
SC8	Mange invitation	This page shows invitation settings
SC9	Send invitation	In this page the user can send invitation to anthor users
SC10	View invitation	In this page the user can read The received invitations
SC11	Mange teams	This page shows invitation settings
SC12	My team	In this the user can control of his team
SC13	Owner sign up	In this page we sign up as owner
SC14	Home play ground owner	The first page that the owner opens after logging in
SC15	Modify playground	In this page the owner can edit data to his playground
SC16	Your playgrounds	This page displays the owner's PlayGrounds available for reservation When it open and when it close?
SC17	Add playground	In this page the owner can add a new palyground

4. Tools

1- intelliJ IDEA 2020.1.1

2- visual Paradigm 16.1

3-starUML

4-Web technologies

5. Ownership Report

Software Design Specification

Item	Owners
Class diagram	Mohamed mahmoud and Abdelrahman fesal
Sequence diagram	George
Interfaces	Nabil and mohamed Ahmed
Implementation	Mohamed mahmoud and Abdelrahman and George
Hosting and documentation	Mohamed mahmoud and Nabil
Screen shots and video	Mohamed mahmoud and mohamed Ahmed and Nabil

Appendix A: Code Listing and Screen Snapshots

The Link to the private repo:

(https://github.com/mohamed-mahmoud377/GOFO-finalProject-CS251-)

And there is an invitation has been sent to **https://github.com/mramly** as a **collaborator** .

The link for the Drive :

(https://drive.google.com/drive/u/1/folders/1BkcddQuz24Us_F3_Pa7Rnc3B-_RAcmMY?fbclid=IwAR3JGV24enq6_CEoc6KPFkEVH8Rt_AYqo5Ljz2EcmS3pOZEiR92unthbsLA

And this **shared drive** has been shared with **m.elramly@fci-cu.edu.eg**.

The link for the explanation video :

(https://drive.google.com/file/d/1JPNneljfN2AjJyvHT9LJsjwTdfh8AOAR/view?usp=sharing)

Software Design Specification

Snapshots from the code running :

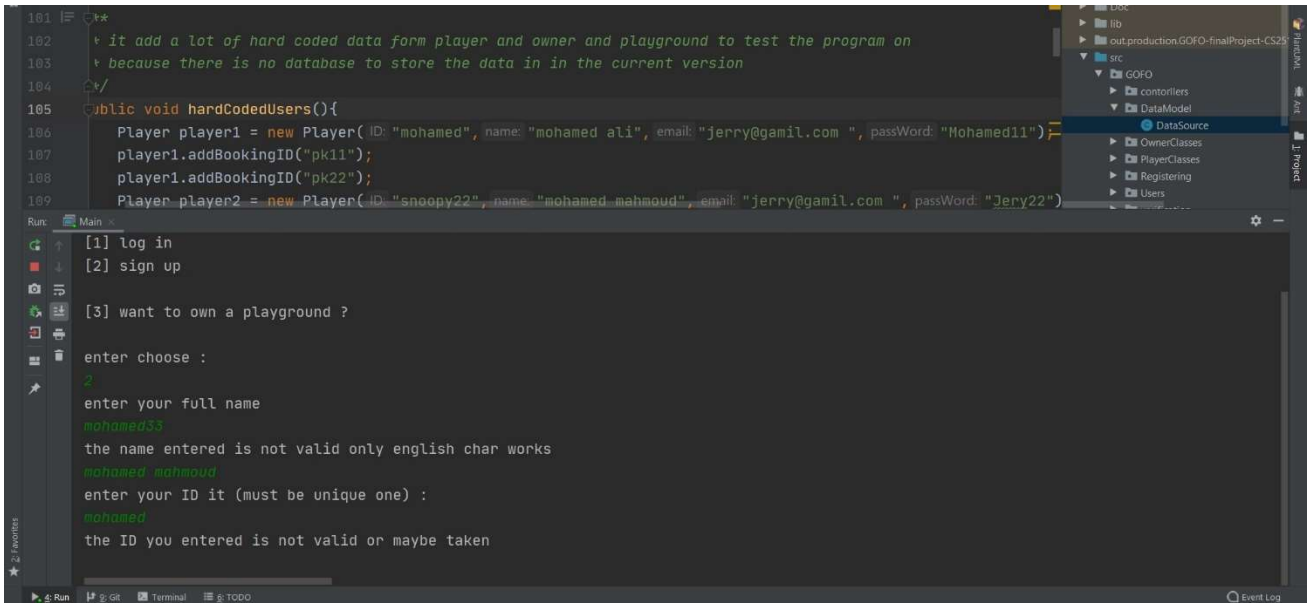
Signing up as player :

1-This is the home screen page of the program gives you the option to log in and sign up or be a play ground owner

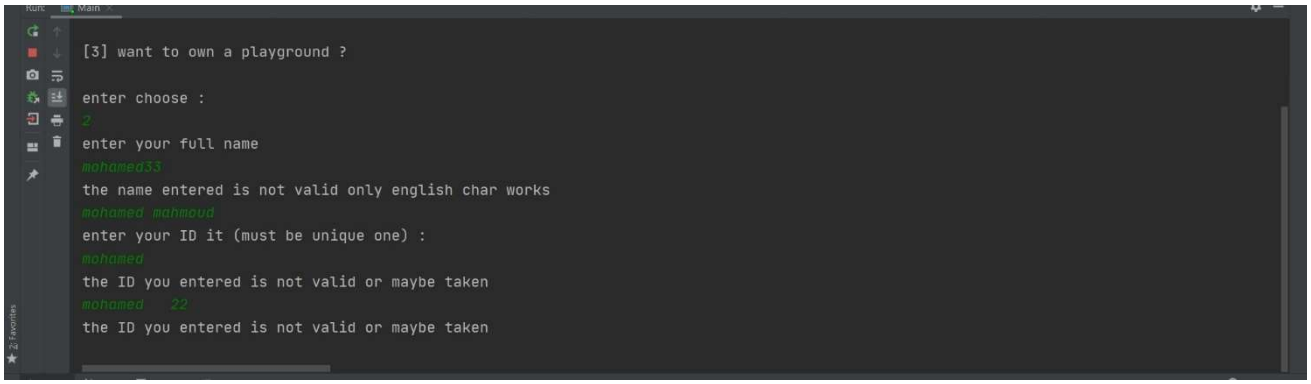
2-and now we sign up as new player and as you see we chose 2 and then asked for the full name and when we tried to put numbers on it we has an error

3-now we entered a suitable full name and has been asked for for an ID and the ID must be unique and as you see here this some hard-coded players and one of them has the some ID as we entered so the program rejected it

Software Design Specification



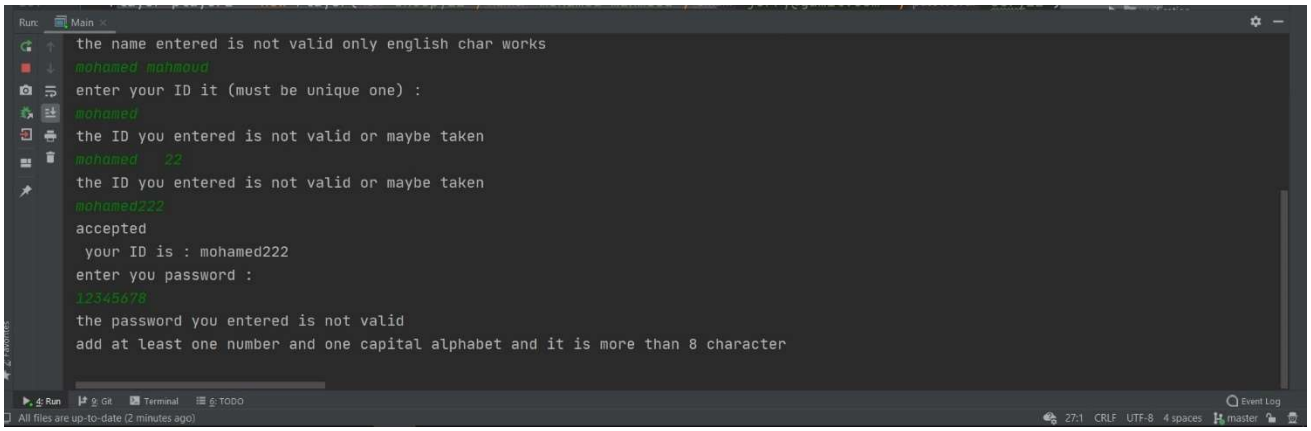
4-and as you see here ID cannot have spaces so I was rejected again



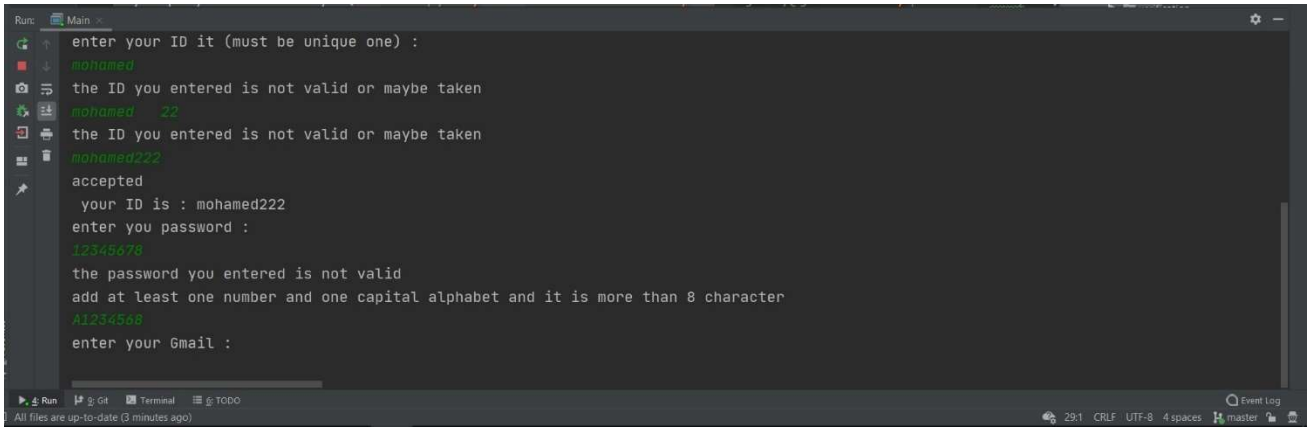
5-now we entered mohamed222 and it is unique and acceptable , and now we are entering the password and as you see only numbers does not work it has to have at least one capital alphabet

Software Design Specification

and one number

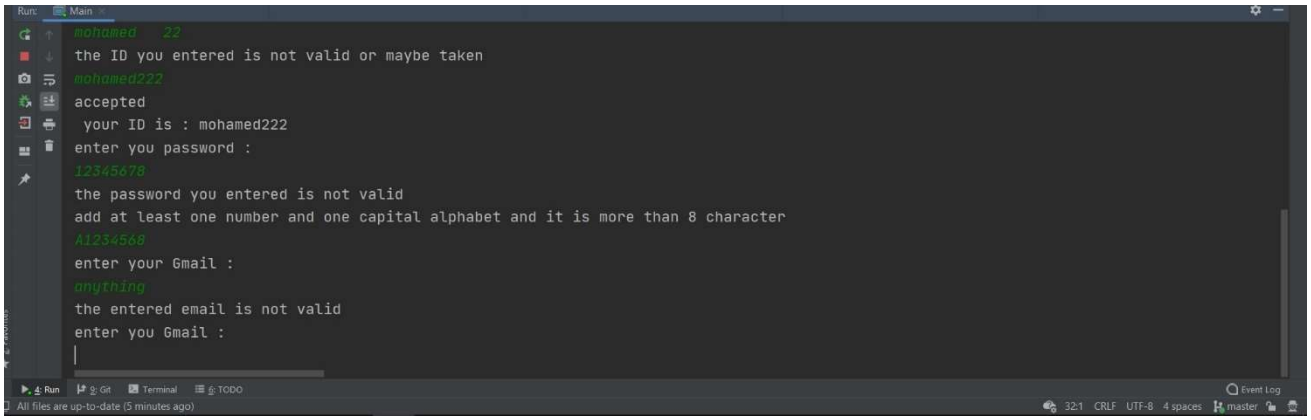


6-and now this is was right password and then the new player has to sign up his email and it only works with Gmail

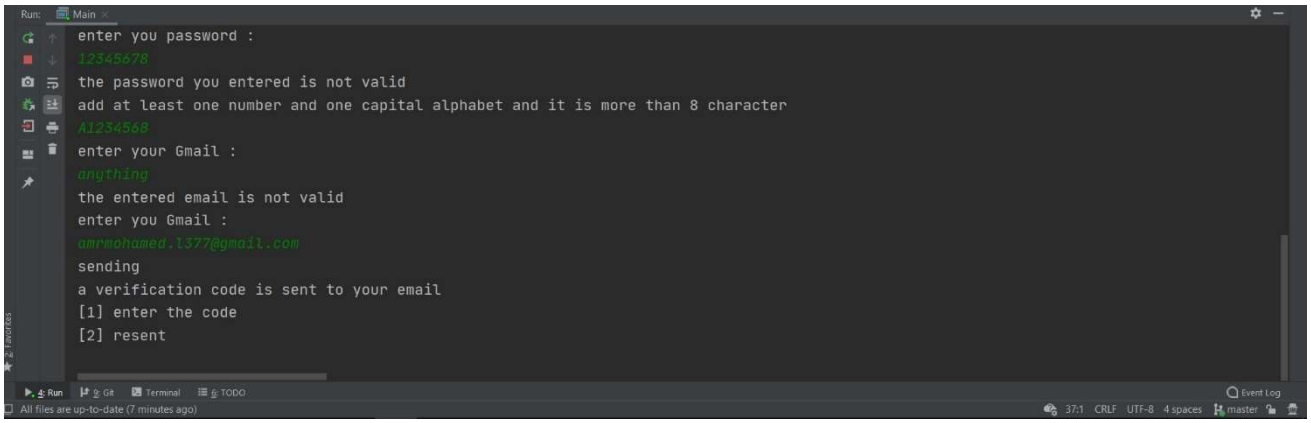


7-if we try to enter any email the program checks of this email can be valid or not and in this case it is not so we have to enter it again

Software Design Specification



8- now we entered a valid Gmail and now a code has been sent to this email and you have to enter it in order to continue

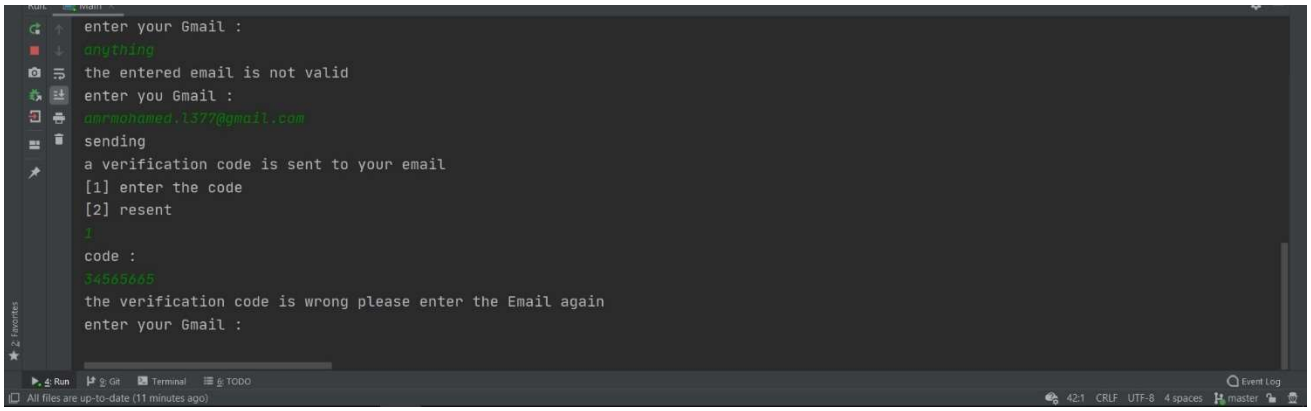


9- and as you see from this screen shot from my phone this how the massage will be send to the User and has the code and this code is auto generated

Software Design Specification

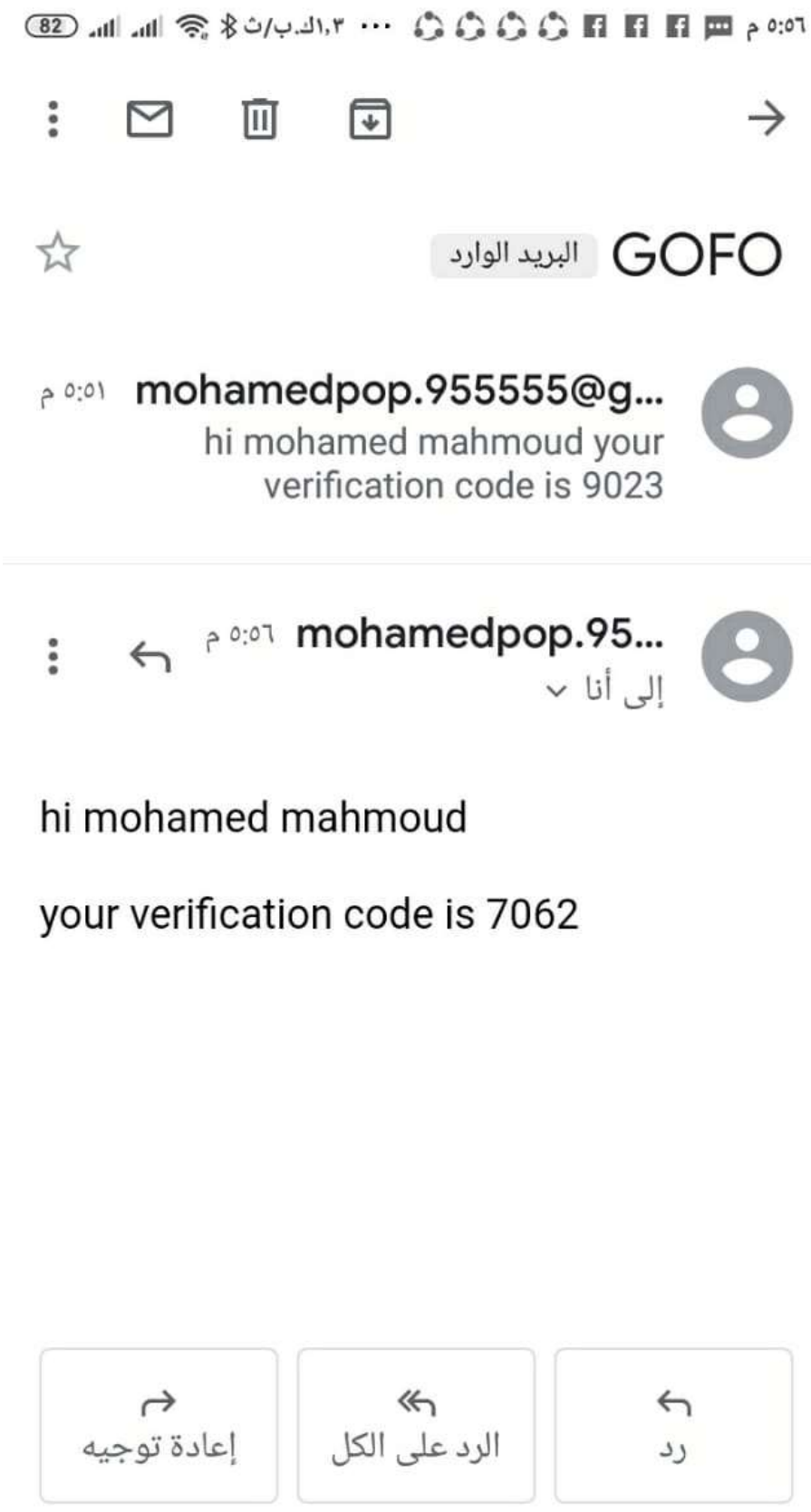
Software Design Specification

10-now I will try to enter the code but I will enter a wrong one not as was sent to me and the program knows that and reject it



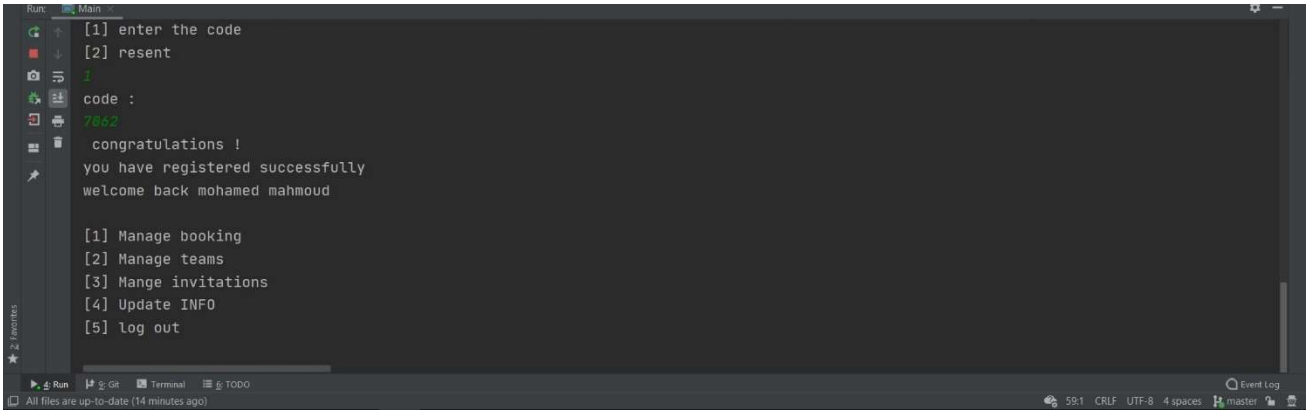
11- and I tried to sent a new one here as you and this is the code that was sent to me

Software Design Specification

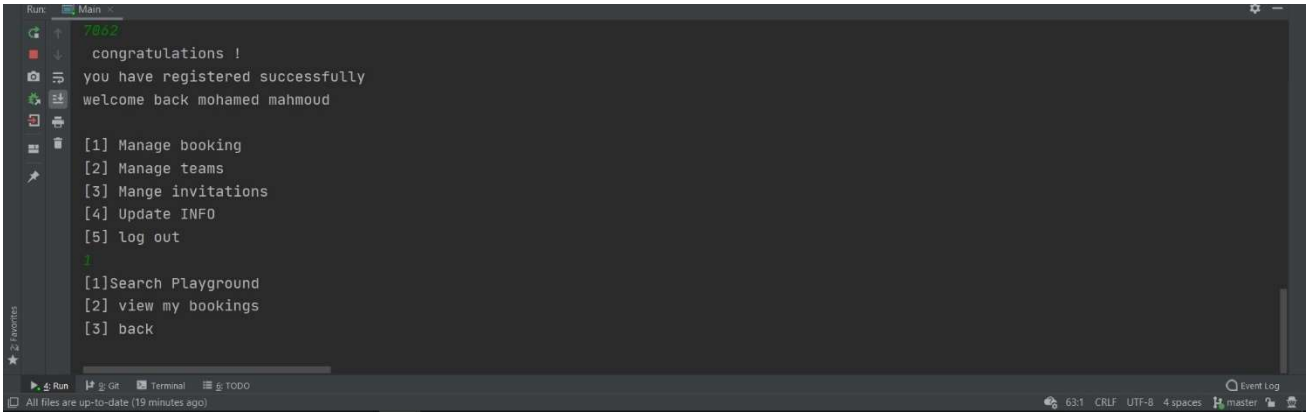


Software Design Specification

12-and here we entered the given code and it was correct and now we registered successfully and this the Player profile here as you see he can manage booking or teams or invitations and update info or even log out

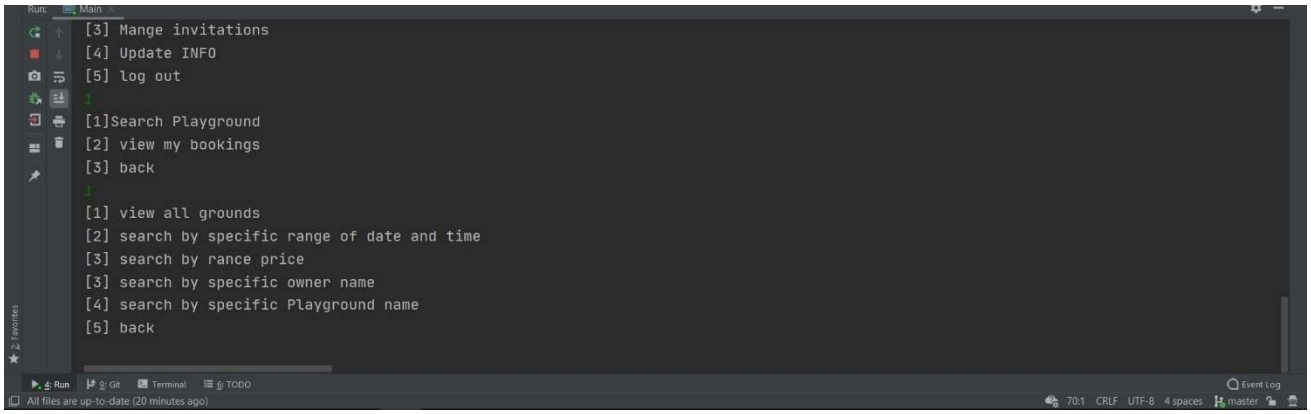


13-and here we choses the manage booking and so we can search for any play ground or view the bookings we had booked and back

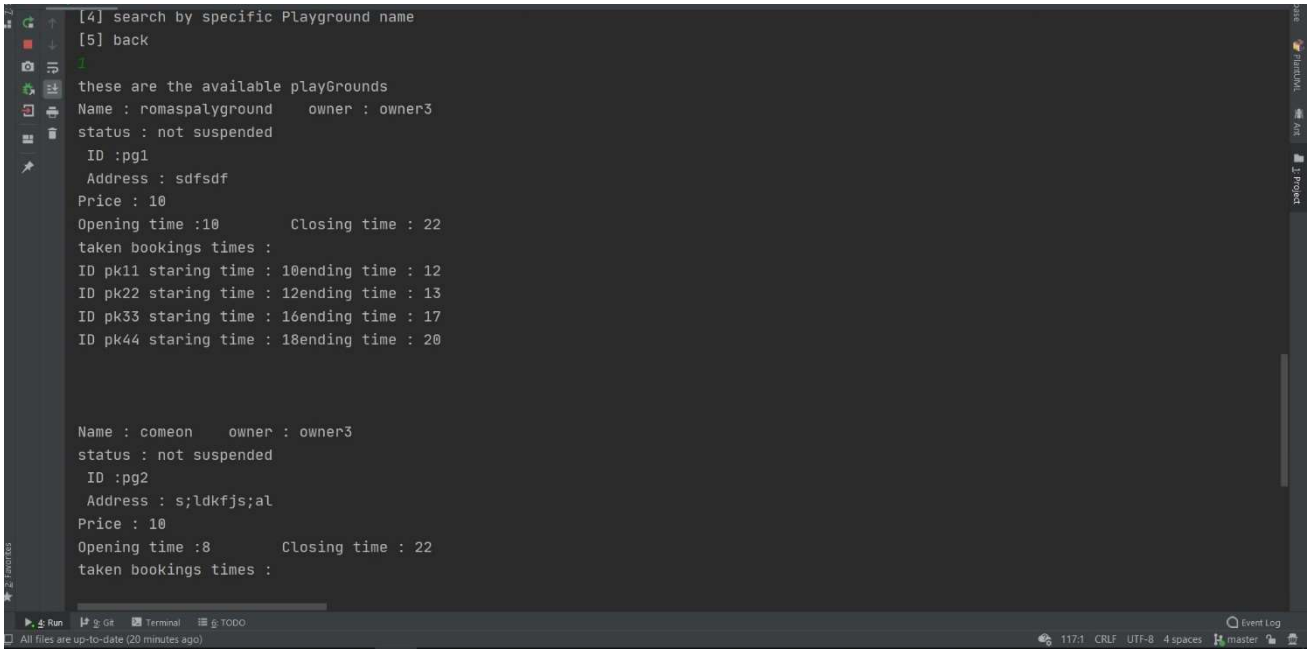


14- and as we see here we got all of this types of searches and you can chose anyone of them

Software Design Specification

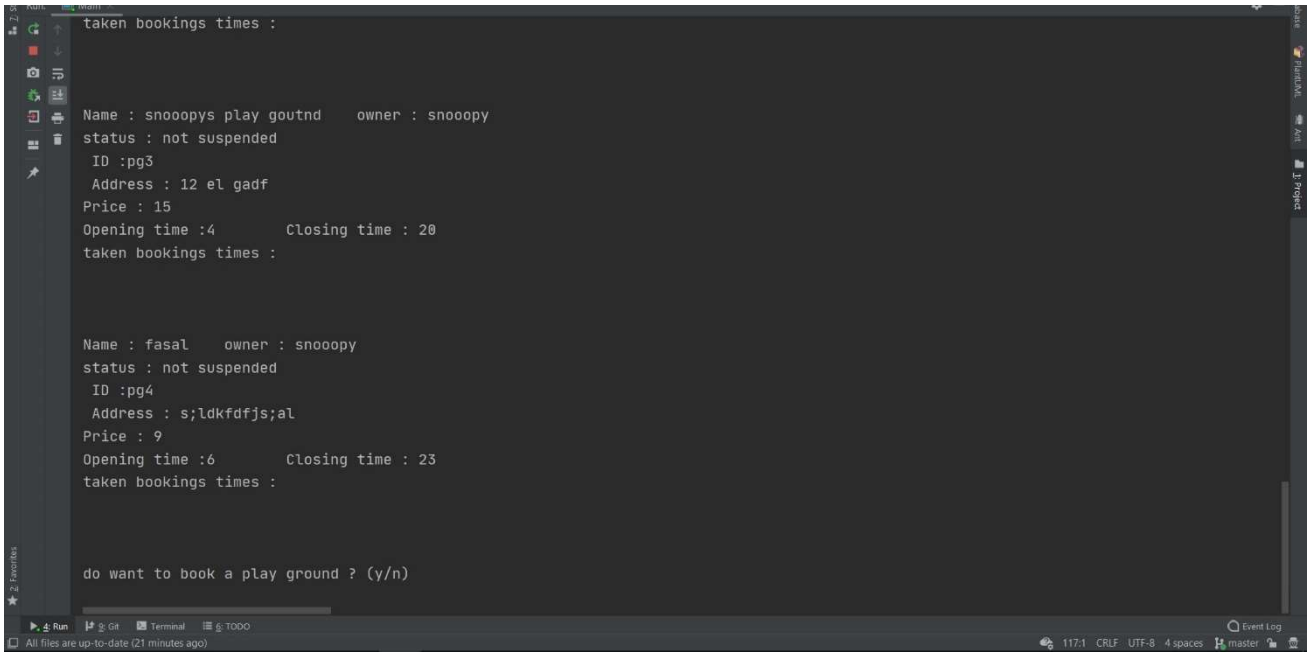


15- and here we chose number 1 to view all the available playgrounds for you to book and all of theses playground are hard coded means that it was added manually to the program bye code to test it out and as you see the only playground that has bookings is the playground with ID :pg1 the first playground in the list



16- and these are the rest of the list and all of the owner of these playground are hard coded too and now the user is asked to book a playground if he want .

Software Design Specification

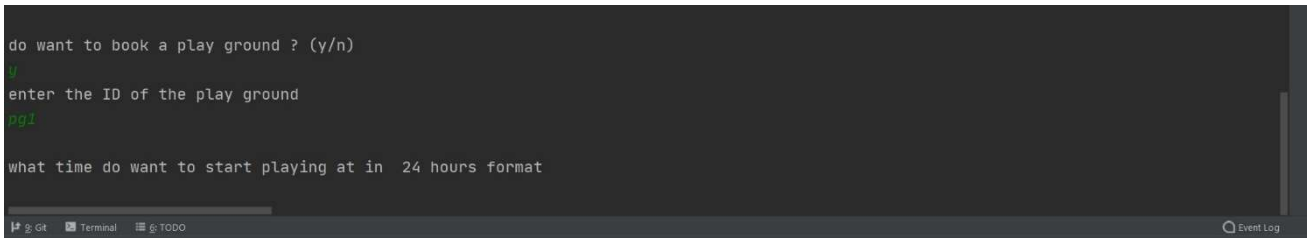


17- and as we see here this is the code for the first playground in the list all of its info were added manually and unfortunately there was not time to add a database for the program.

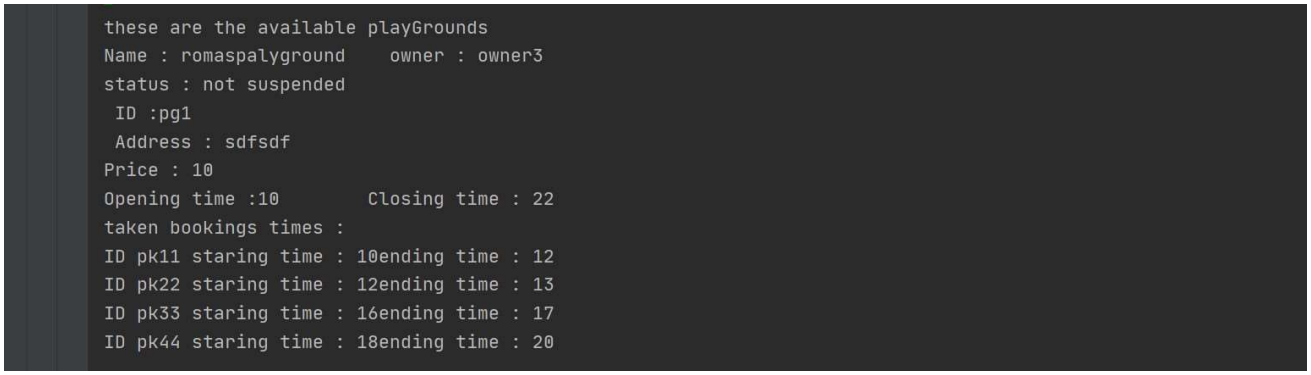
```
Ground ground1 = new Ground( ID: "pg1", name: "romaspalyground", description: "aiaa jalsdkfj;aslkdfjajl;", addr
        ,owner2.getID(), openingTime: 10, closingTime: 22, price: 10, isSuspended: false);
Booking booking1 = new Booking( startingDate: 10, endingDate: 12, ID: "pk11", playerID: "jerry11");

Booking booking2 = new Booking( startingDate: 12, endingDate: 13, ID: "pk22", playerID: "jerry11");

Booking booking3 = new Booking( startingDate: 16, endingDate: 17, ID: "pk33", playerID: "snoopy22");

Booking booking4 = new Booking( startingDate: 18, endingDate: 20, ID: "pk44", playerID: "snoopy22");
ground1.addBooking(booking1);
ground1.addBooking(booking2);

ground1.addBooking(booking3);
```

18-and now we will book the pg1 playground so we were asked to enter the ID for playground we want to book so we entered it and now we should enter the staring time for the User to book.

Software Design Specification



19- and these as we see all the taken times from the playground .



20- and now we are trying to enter anything any numbers here as we see and has been rejected it has to be in 24 format .

21- and now we try to enter a time which was already taken so it was rejected to .

Software Design Specification

```
what time do want to start playing at in  24 hours format
43
hat time do want to end the match at in  24 hours format
65
the time you entered is wrong or taken try again
what time do want to start playing at in  24 hours format
11
hat time do want to end the match at in  24 hours format
12
the time you entered is wrong or taken try again
what time do want to start playing at in  24 hours format
```

22-and now the playground has been booked successfully we enter a time that is not taken and valid

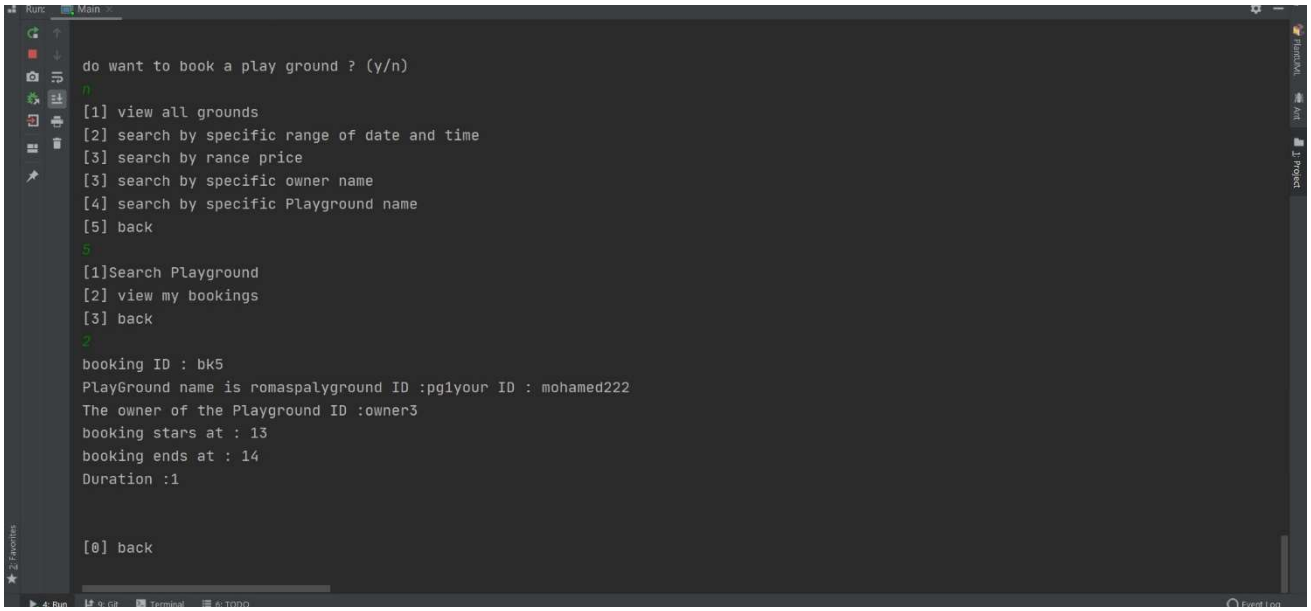
```
what time do want to start playing at in  24 hours format
43
hat time do want to end the match at in  24 hours format
65
the time you entered is wrong or taken try again
what time do want to start playing at in  24 hours format
11
hat time do want to end the match at in  24 hours format
12
the time you entered is wrong or taken try again
what time do want to start playing at in  24 hours format
13
hat time do want to end the match at in  24 hours format
14
you has booked the playground successfully
these are the available playGrounds
Name : romaspalyground    owner : owner3
```

23- and now we will not book another playground and get back.

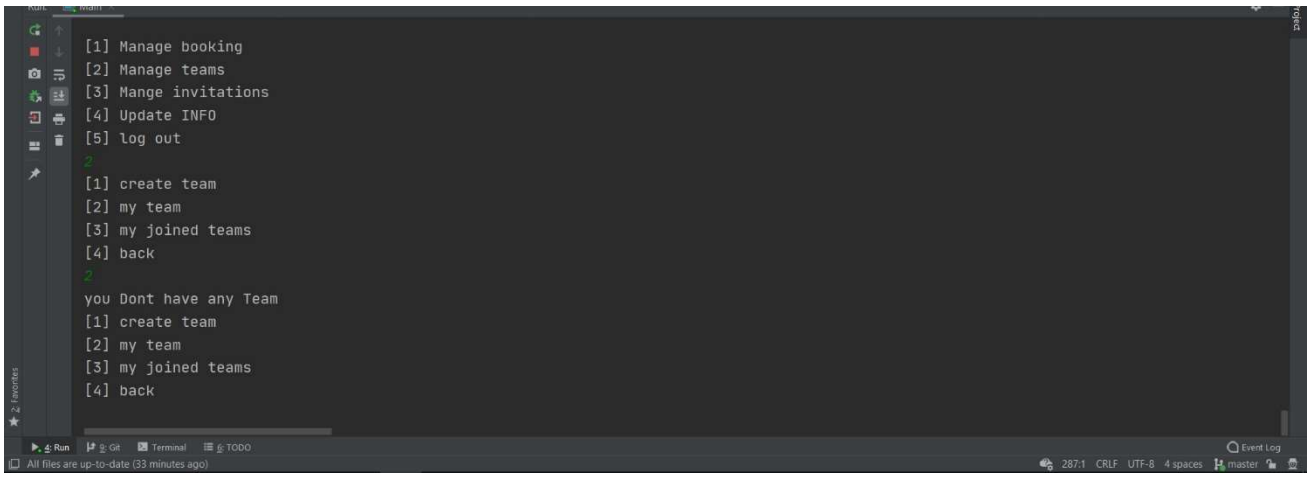
```
do want to book a play ground ? (y/n)
n
[1] view all grounds
[2] search by specific range of date and time
[3] search by rance price
[3] search by specific owner name
[4] search by specific Playground name
[5] back
```

24- and now the go back and open view booking and here we go the booking that we just booked has been added there and with all the info needed .

Software Design Specification



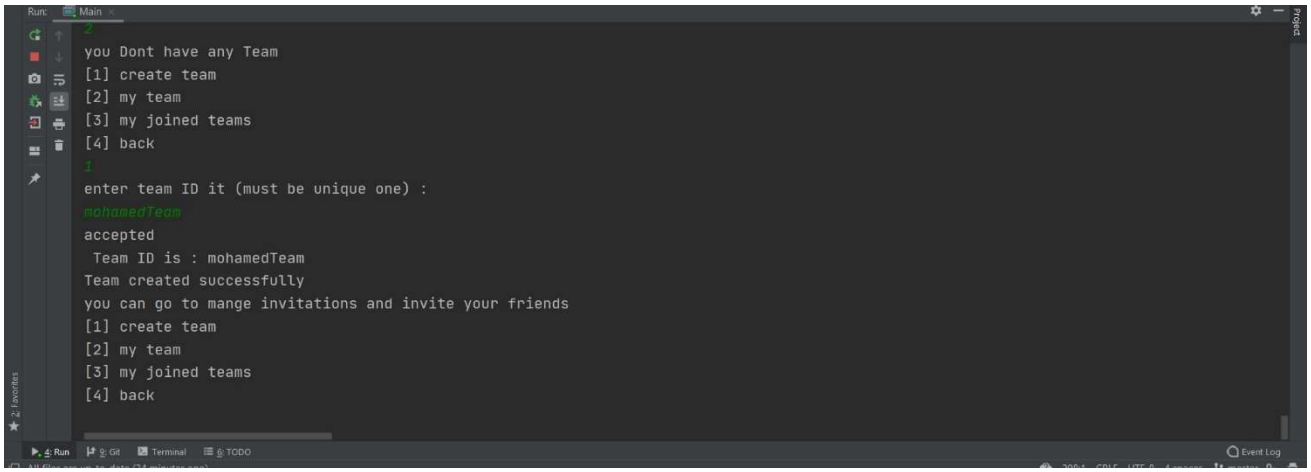
25- and now we are heading to manage teams and then my team and there is not team yet ! we have not create one yet.



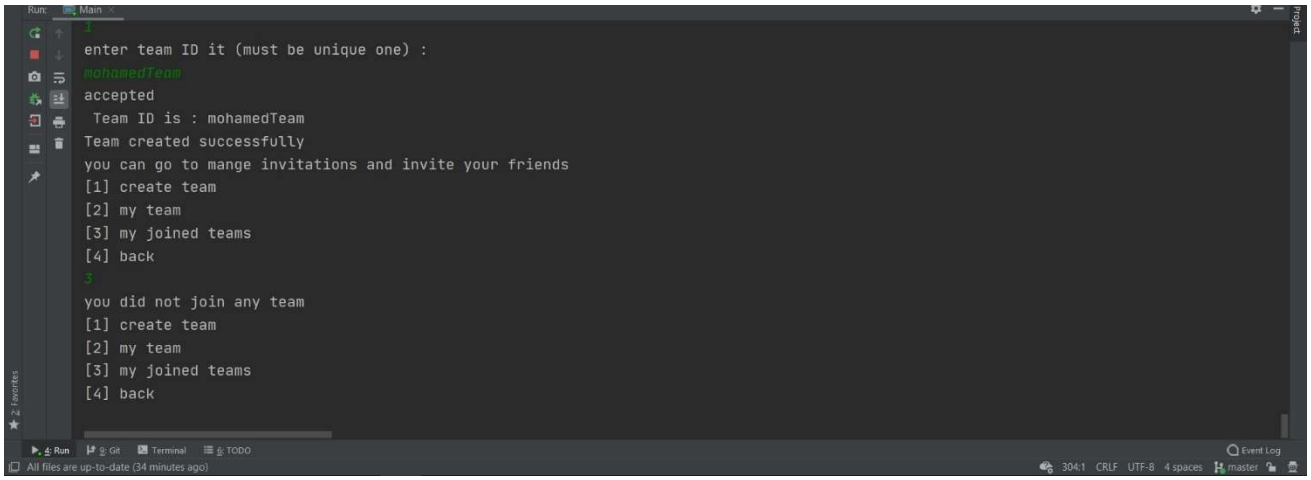
26- so now we will create team here as we see and then we were asked to enter the team ID and

Has to be unique and by the way any ID in the whole program for anything must be unique and it is.

Software Design Specification

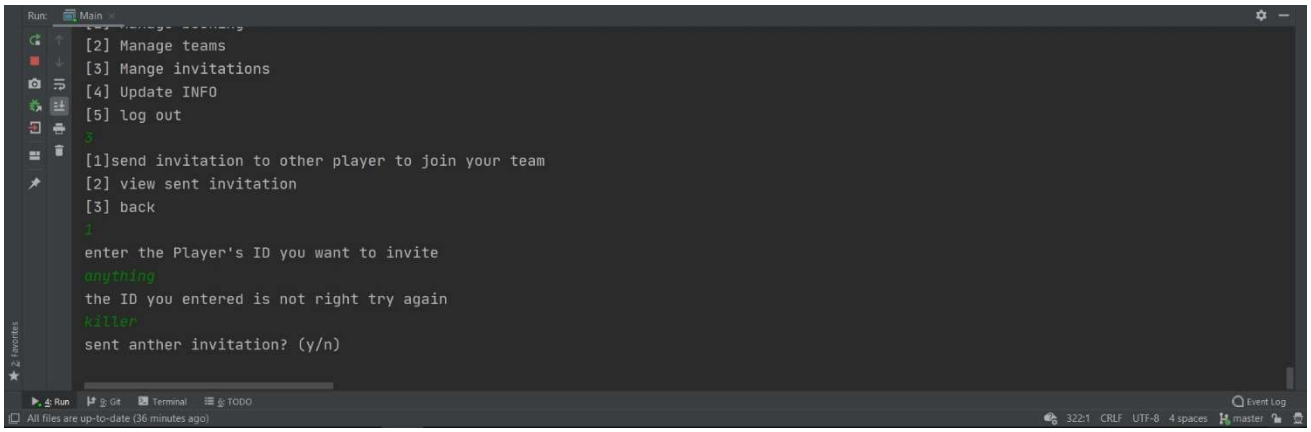


27- and now the team were created successfully and the program telling us if we want to invite any friends it has to be done in manage invitations and as we see too we have not joined any team too so the my joined team is empty .



28- and now we entered manage invitations and then send invitation and then we were asked to enter the player ID we want to invite and when we tried to enter anything in was rejected because it is an valid player ID but when we entered killer I was accepted because this player is already there in the hard coded players .

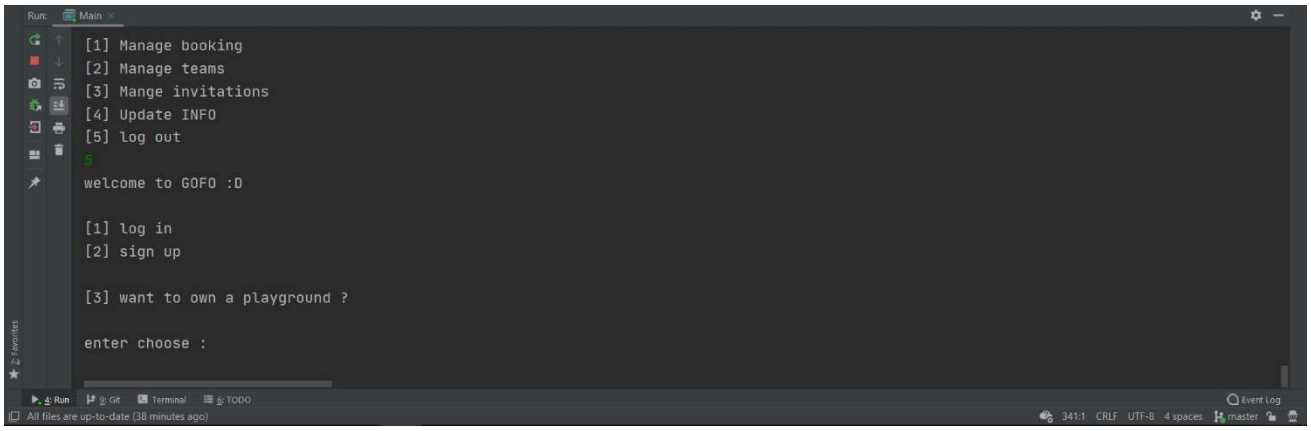
Software Design Specification



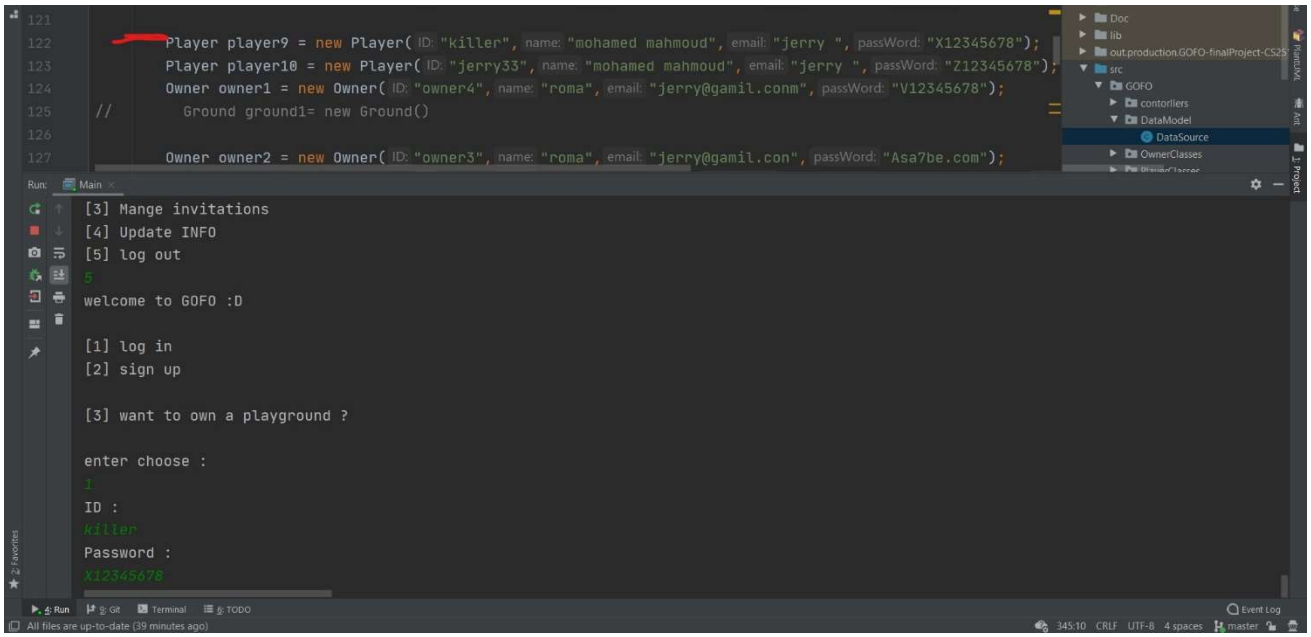
29-and that was enough to sent to killer so we said no but we can sent as many invitations as we want and then we entered a massage to be attached with the invitation.

30-and now we logged out to sign up with the killer player account to see the sent anvitation.

Software Design Specification

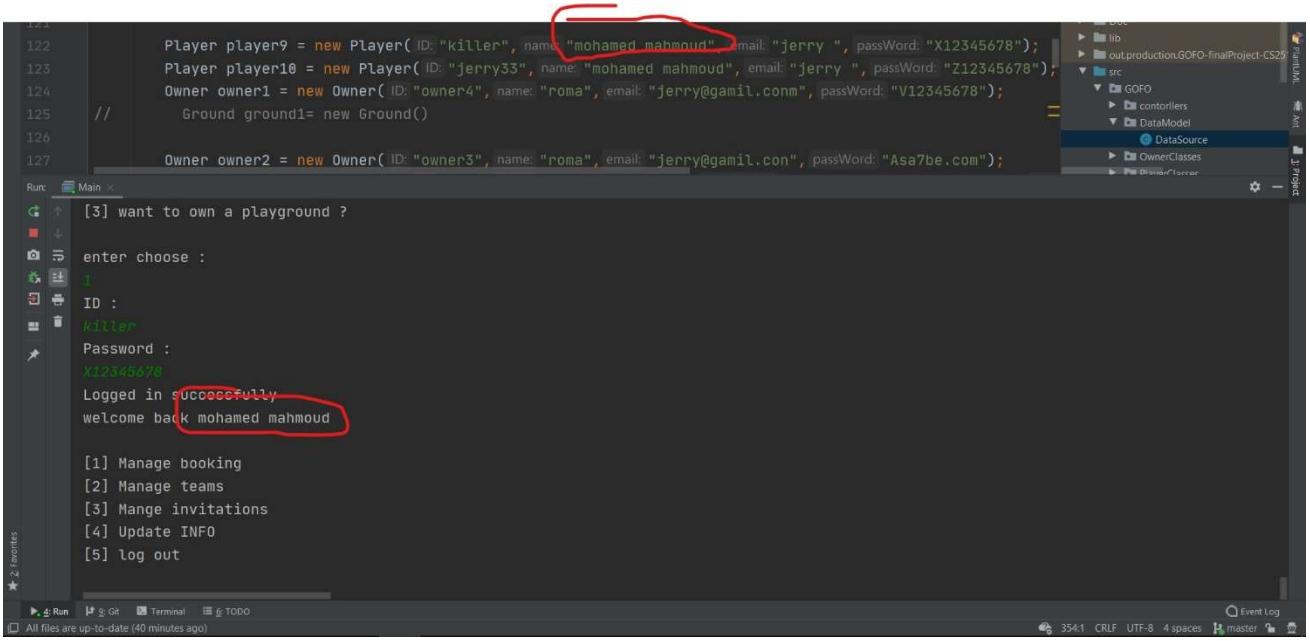


31- and here is the Killer player info from the code and we all sign up with to get to his profile

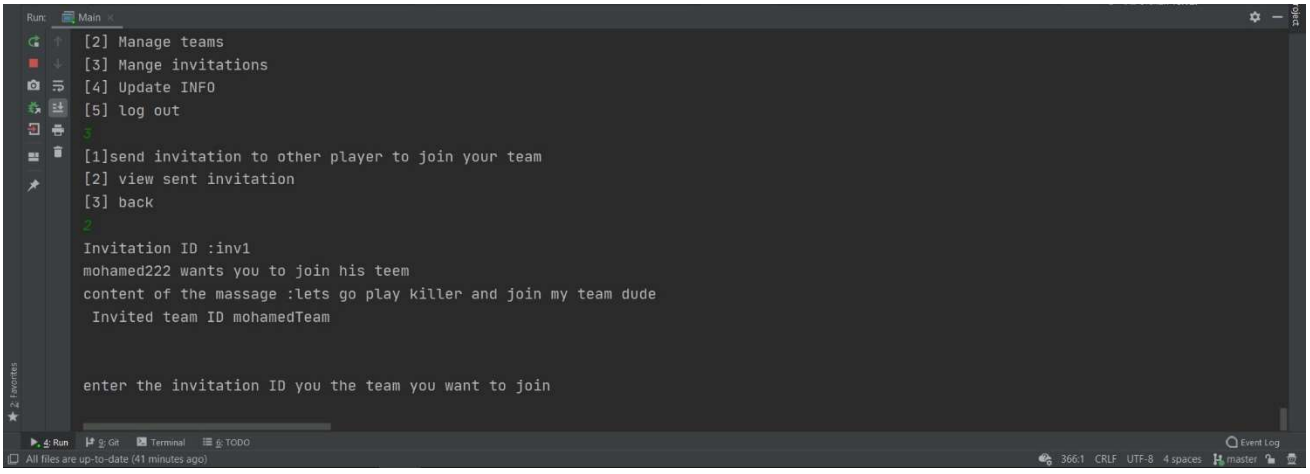


32- and now the entered his profile and as we see the same name appearing in the profile which means that we has logged in successfully to the account .

Software Design Specification

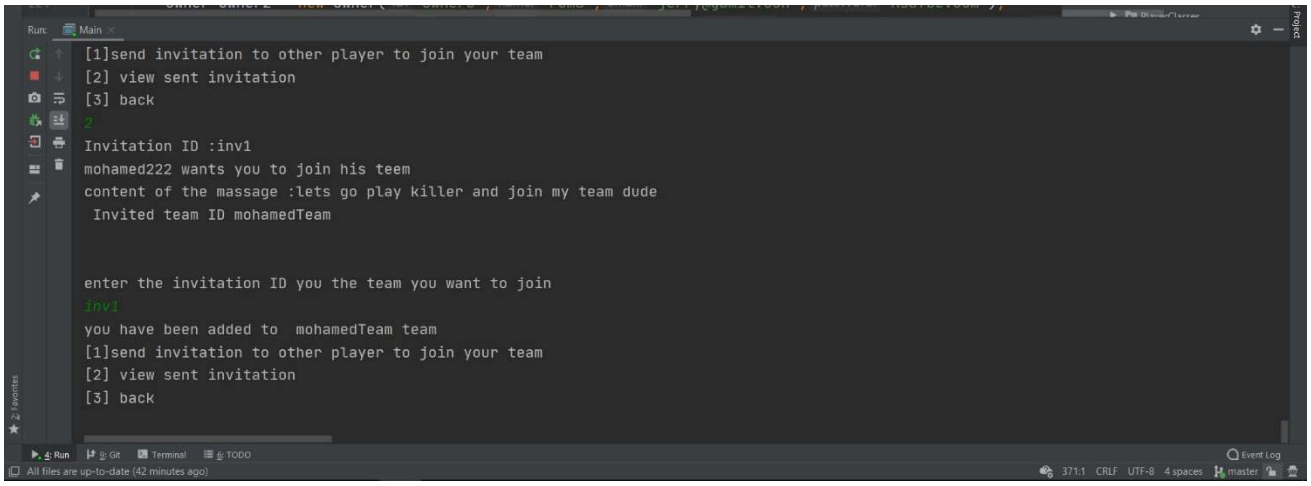


33- and here we go the invitation is here and were sent successfully to killer player with the same massage we sent it with



34- and now we will accept this invitation by entering its ID which is by the way auto generated which means for sure every invitation has a different ID form the other one.

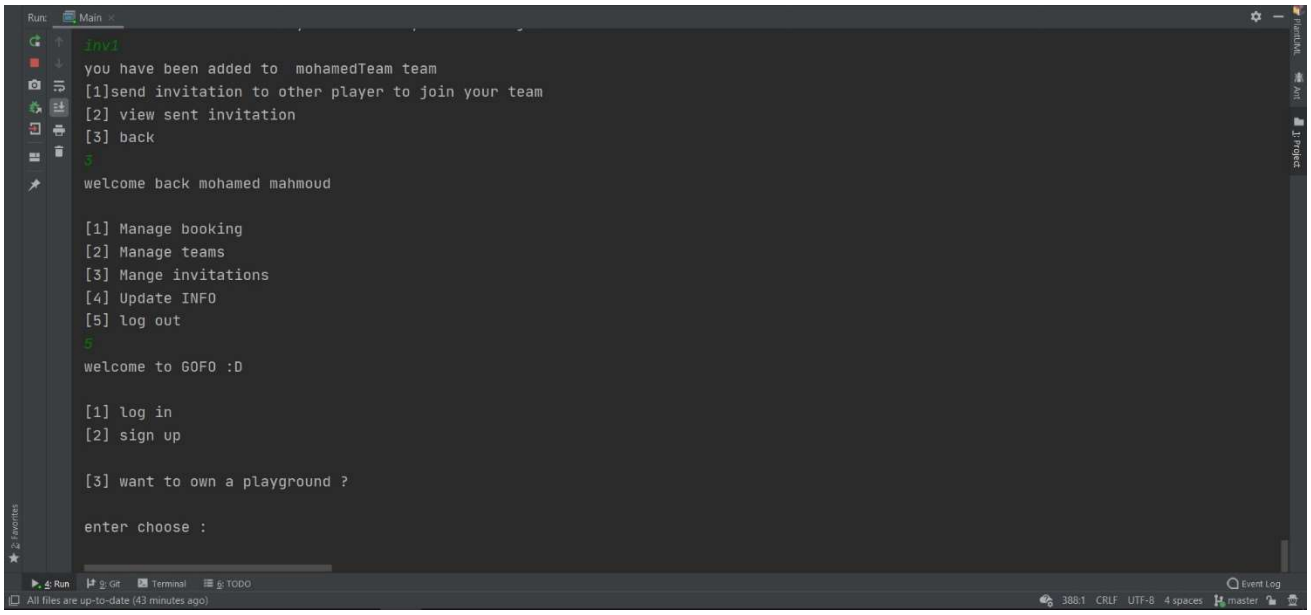
Software Design Specification



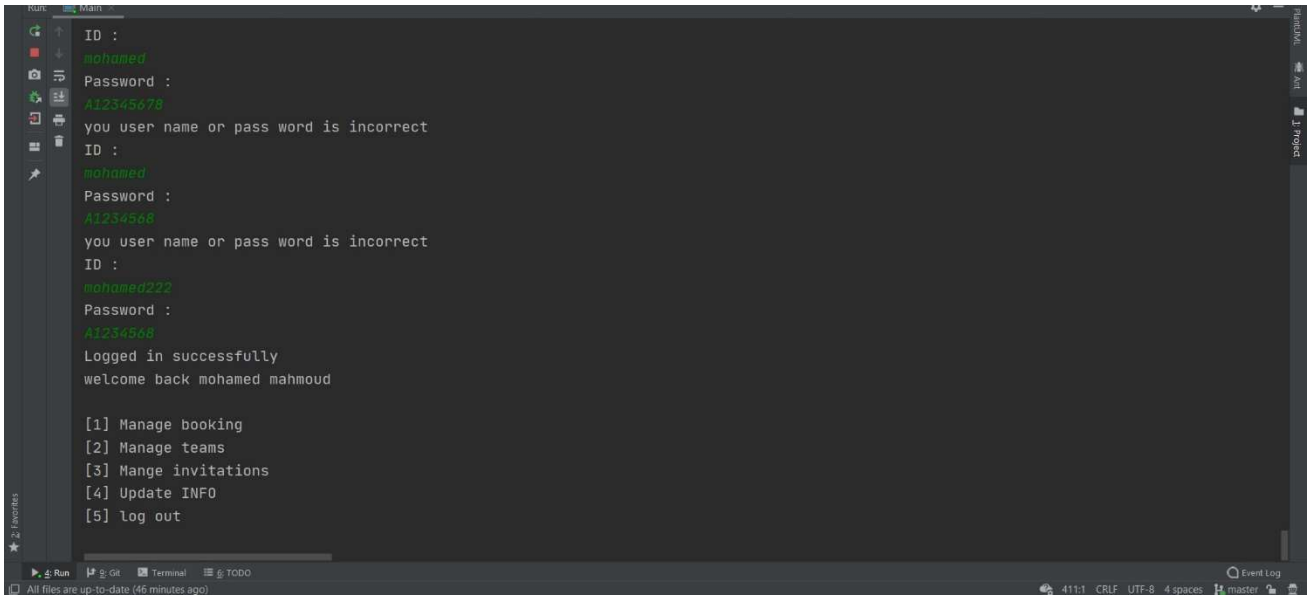
35- and hello we entered my joined teams here is the team which we were created with mohamed222 and the team ID which we provided too and now killer has joined it and you can has till 8 players in your team .

36- and now we logged out from killer profile and we will hit back to mohamed222 which is the player that we have just signed up with to see logging in with a brand new profile and also to check out the team and make sure that killer was added successfully .

Software Design Specification

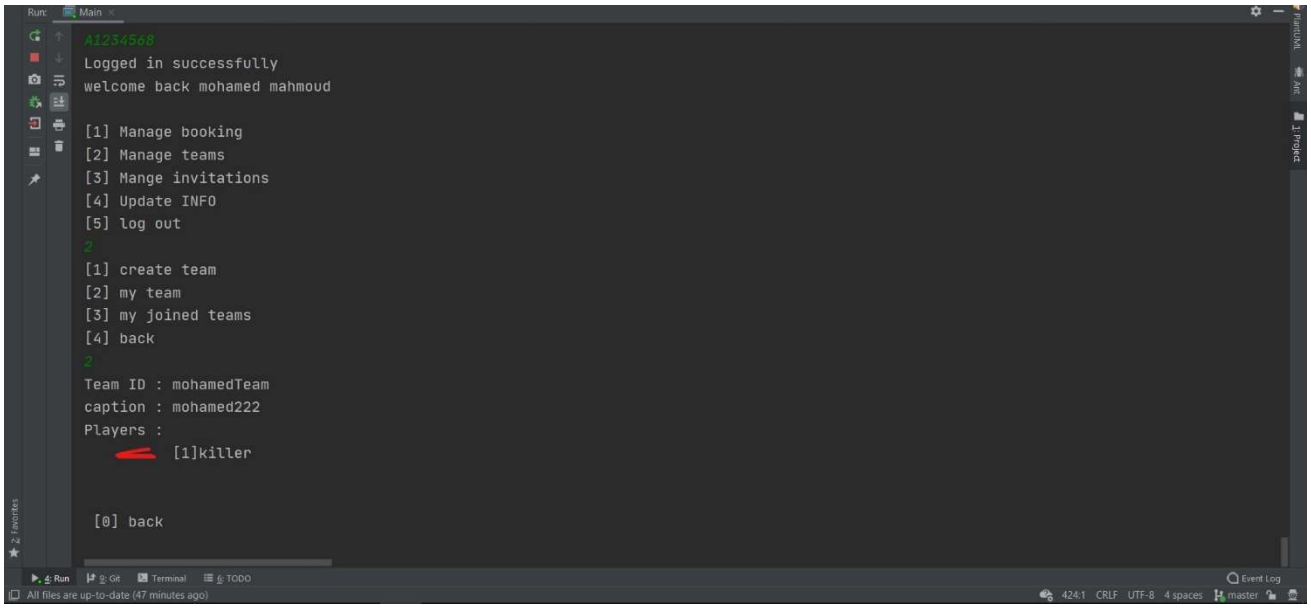


37-and as we here when I was logging in I almost forgot the ID for the profile and I was not able to log in but here we go I got it right and here we are again in the profile so the logging in working just fine as we see.

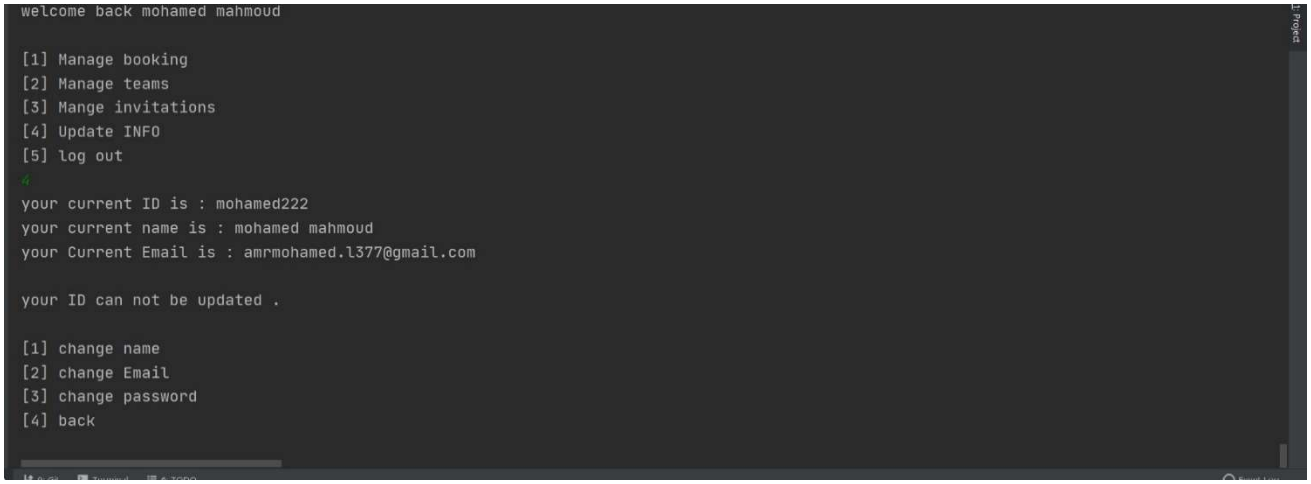


Software Design Specification

38-and now we are entering manage teams and then my team and here he is the killer player is added successfully to our team.

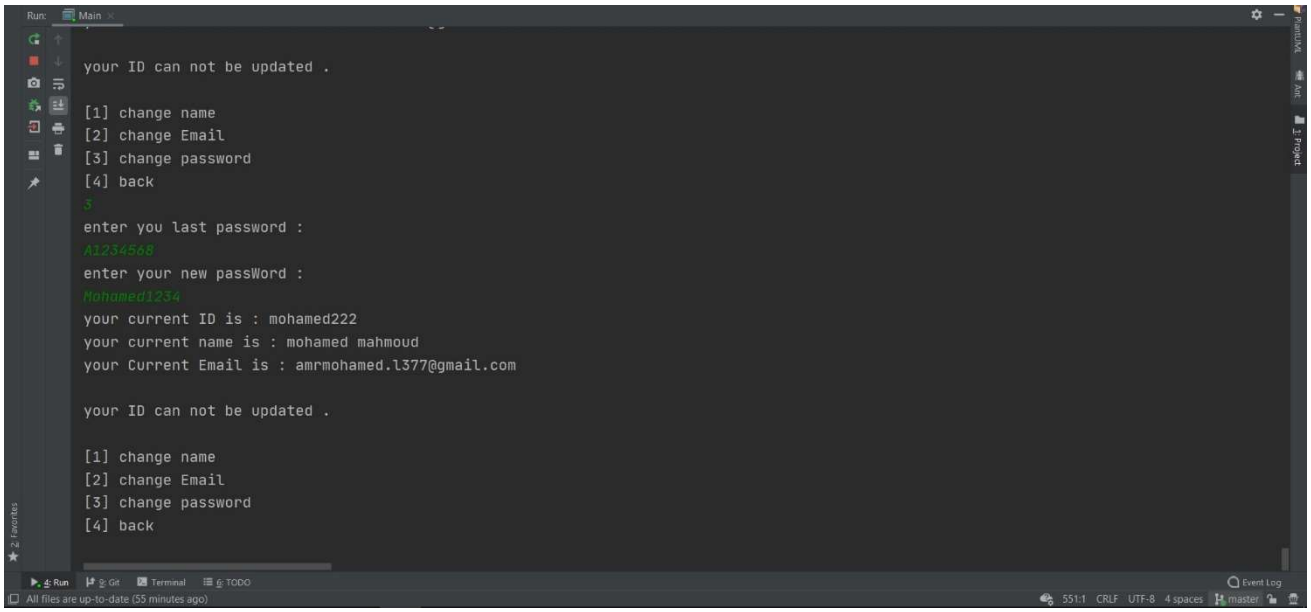


39- and now we also can update info and ID is not changeable



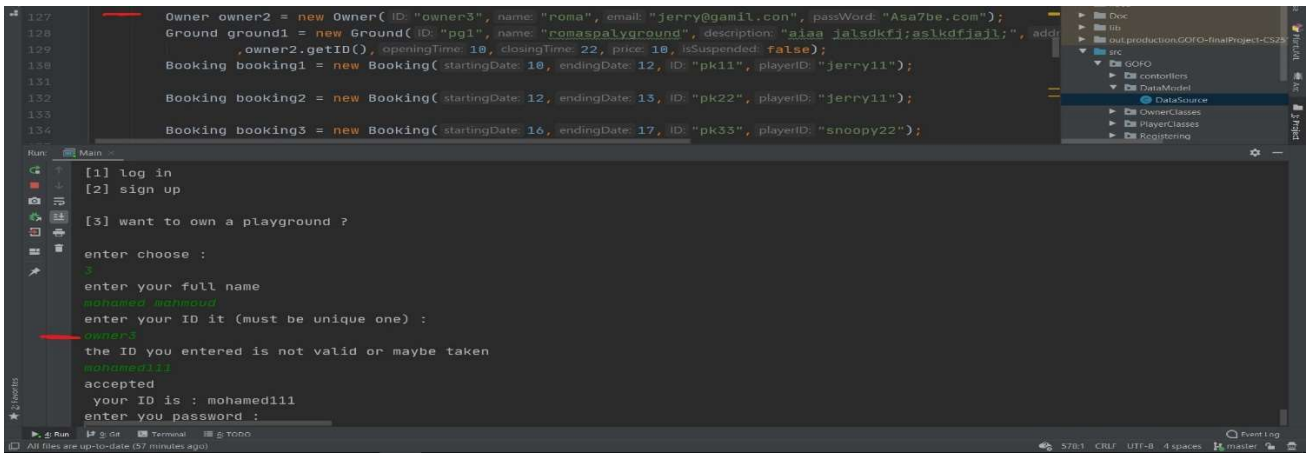
Software Design Specification

40- and here we are trying to update the password for example and it works just fine.



Signing up as Owner :

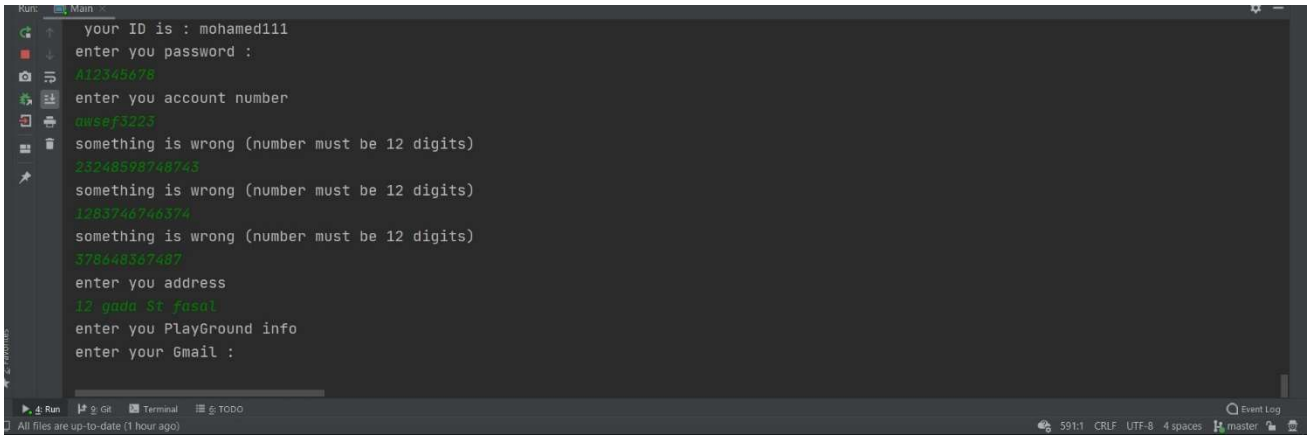
41- and now we got out of the player profile and lets create a owner profile this time and here we entering our full name and then the ID and as we see I entered a ID that was already exist and was rejected and then entered mohamed111 and got accecpted.



Software Design Specification

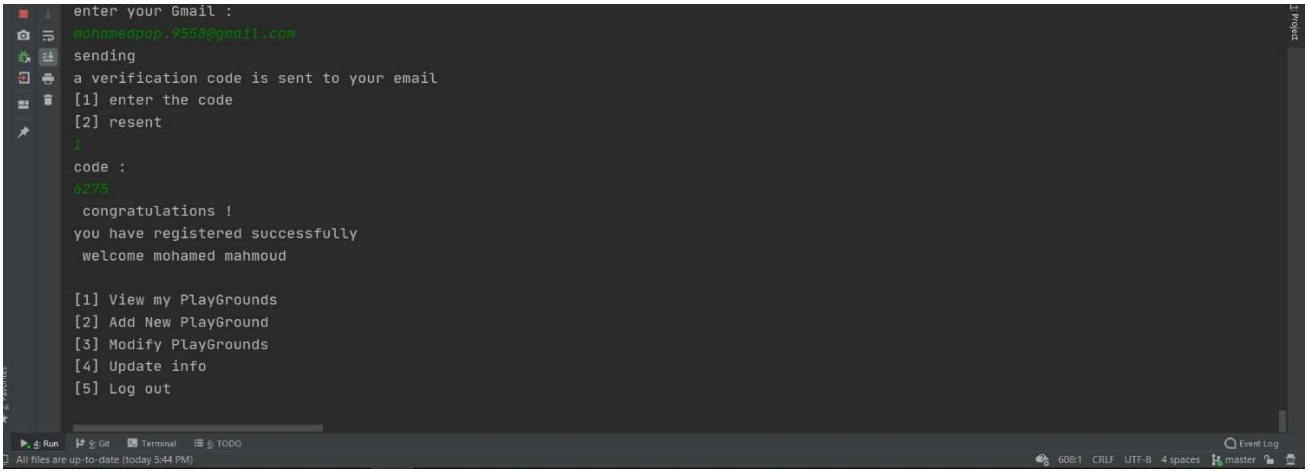
42- and now the owner has to enter is bank account number and it gives an error because it has to be 12 digits

43- it was hard to enter random 12 for me :D and then you get asked about your address and then the some as the player will enter the email and then get the validation code

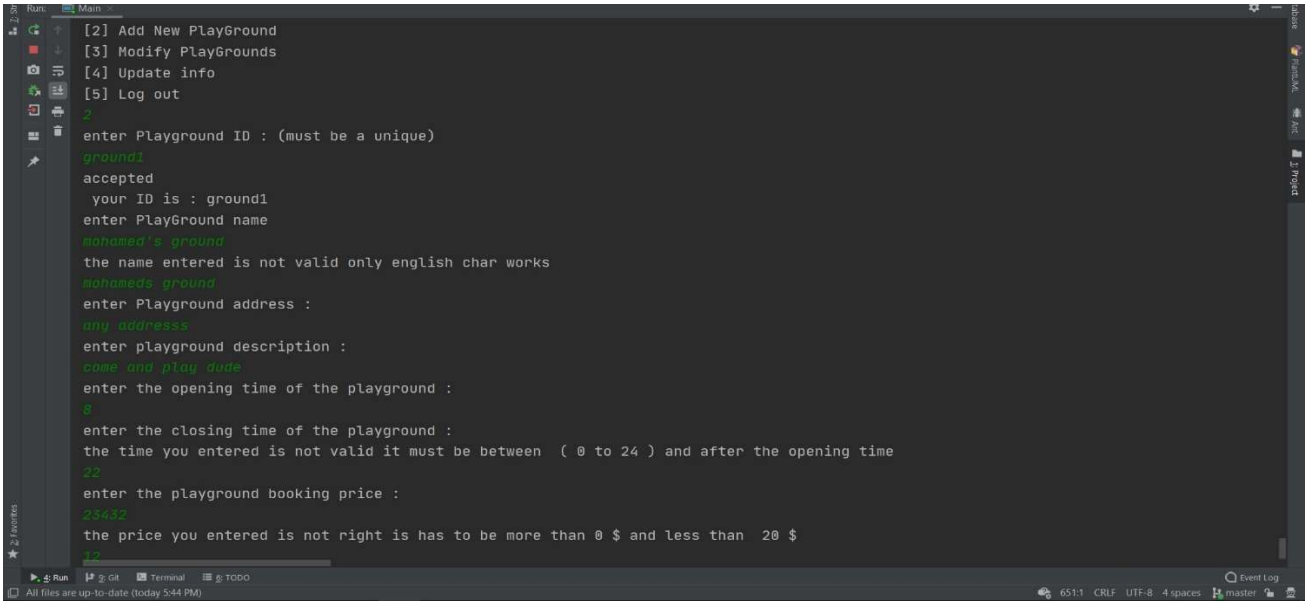


Software Design Specification

44-and then the email were confirmed and the account has been signed up successfully and this the Owner profile be like you can view your playgrounds and add a new one or modify



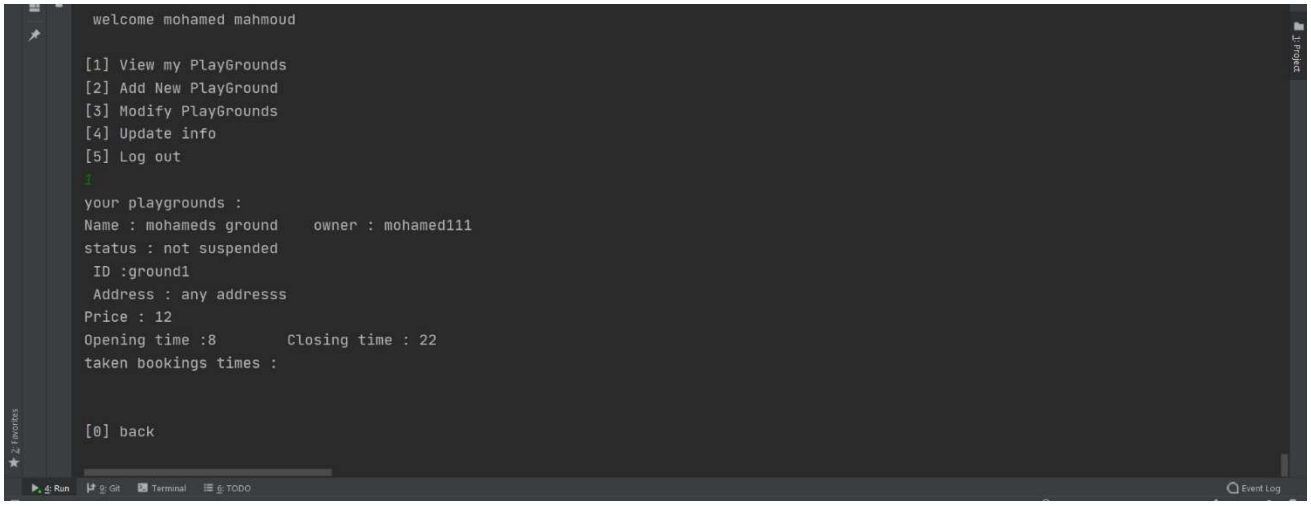
45-and now we are adding a new playground and entering the ID we want it has to be unique too and then the name of the playground and then the address and then the opening time and the closing time of the playground and the some rules apply when we tried to add a new booking with the play it has to be in the 24 format and then the price it has to be form 1$ to 20$ this is the limit price for a playground



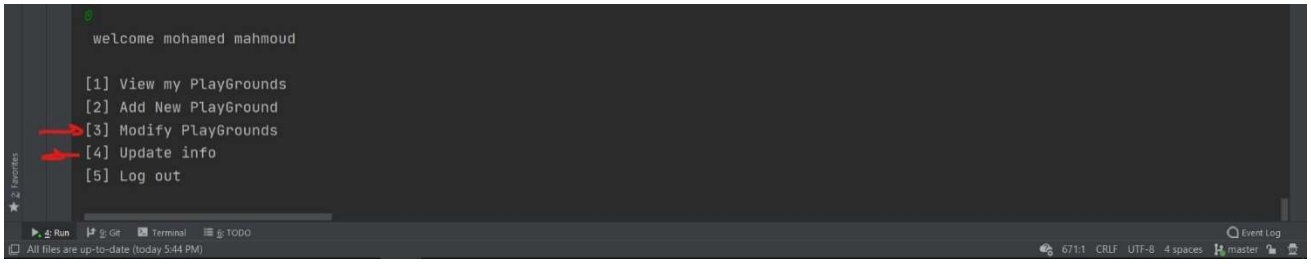
Software Design Specification

46-and now when we go to view my playground here it is the playground that we have just added

And as you see it is not suspended actually it has to be suspended but it is not as default for now for testing purpose only.

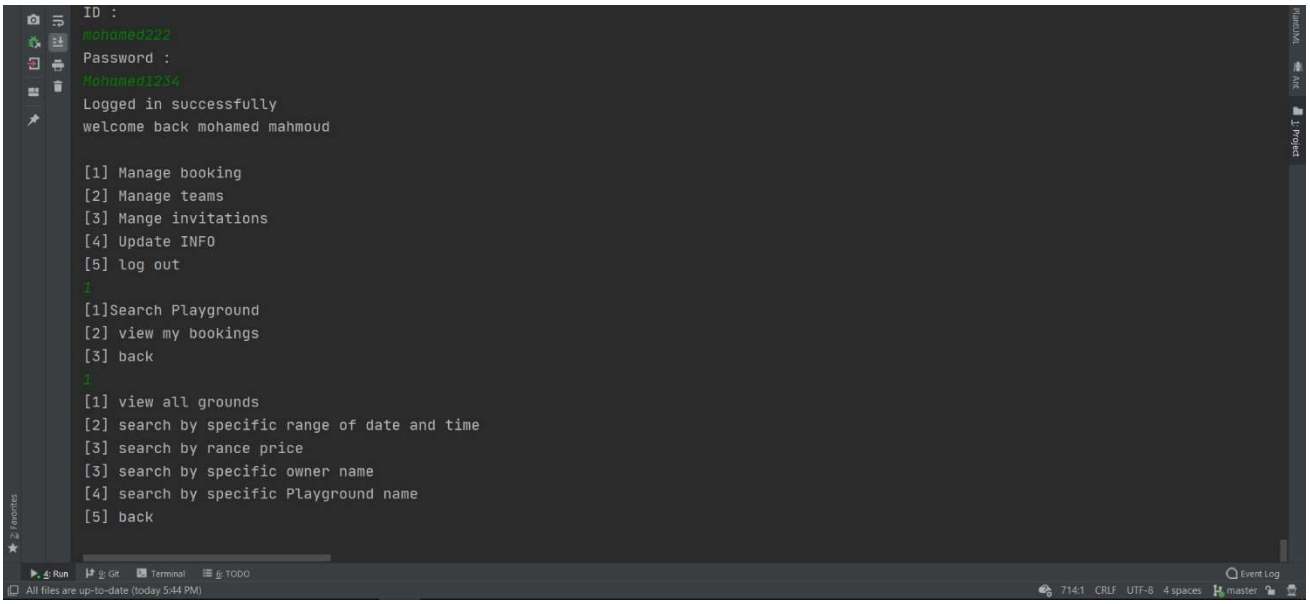


47- and of course we can modify any playground or update info the same way we do in the player profile.

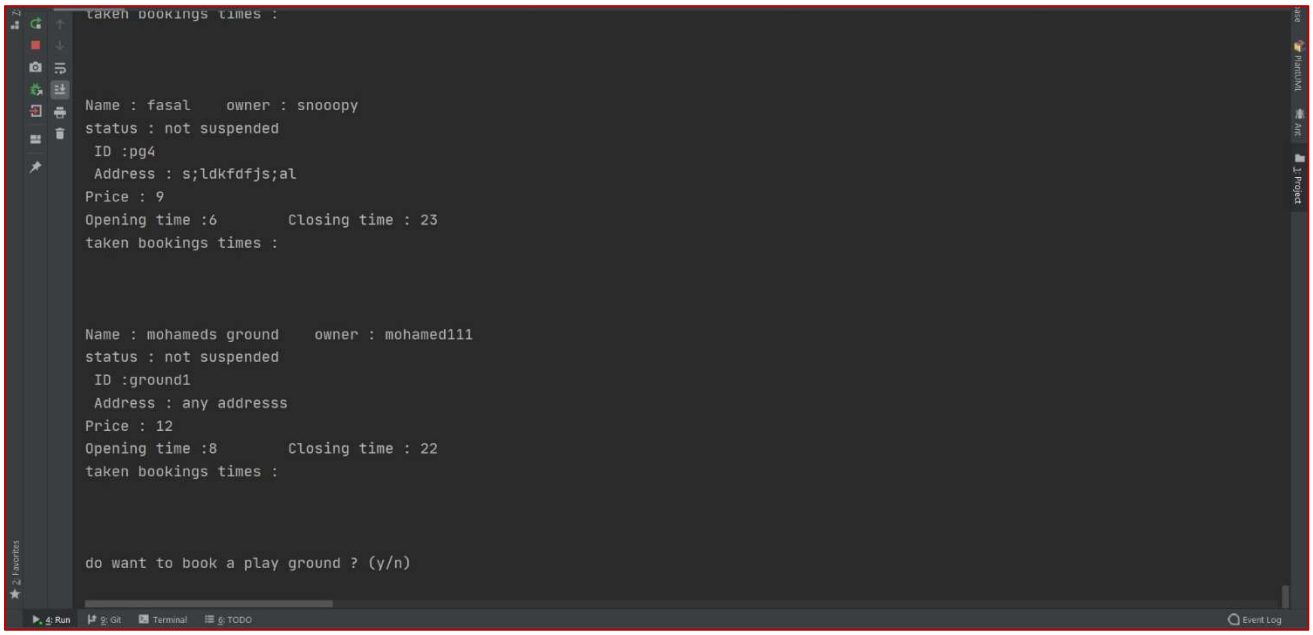


48-and now let's go and log in again to mohamed222 player profile that we created earlier to check out the new playground that we has just added.

Software Design Specification



49-and here it is the last playground here is the play ground that we has just created is avilabale to the players

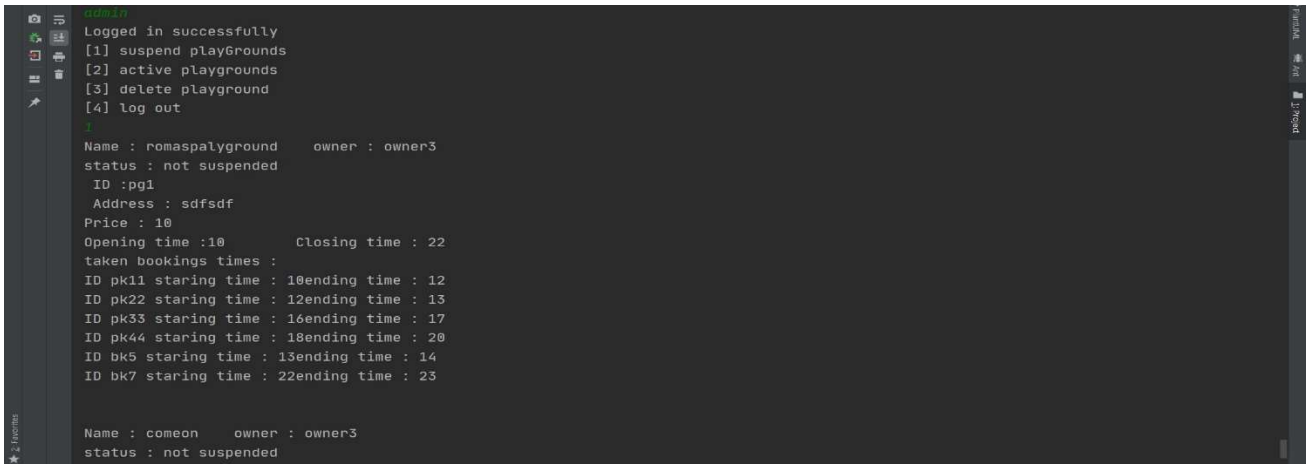


Software Design Specification

Signing up as Admin:

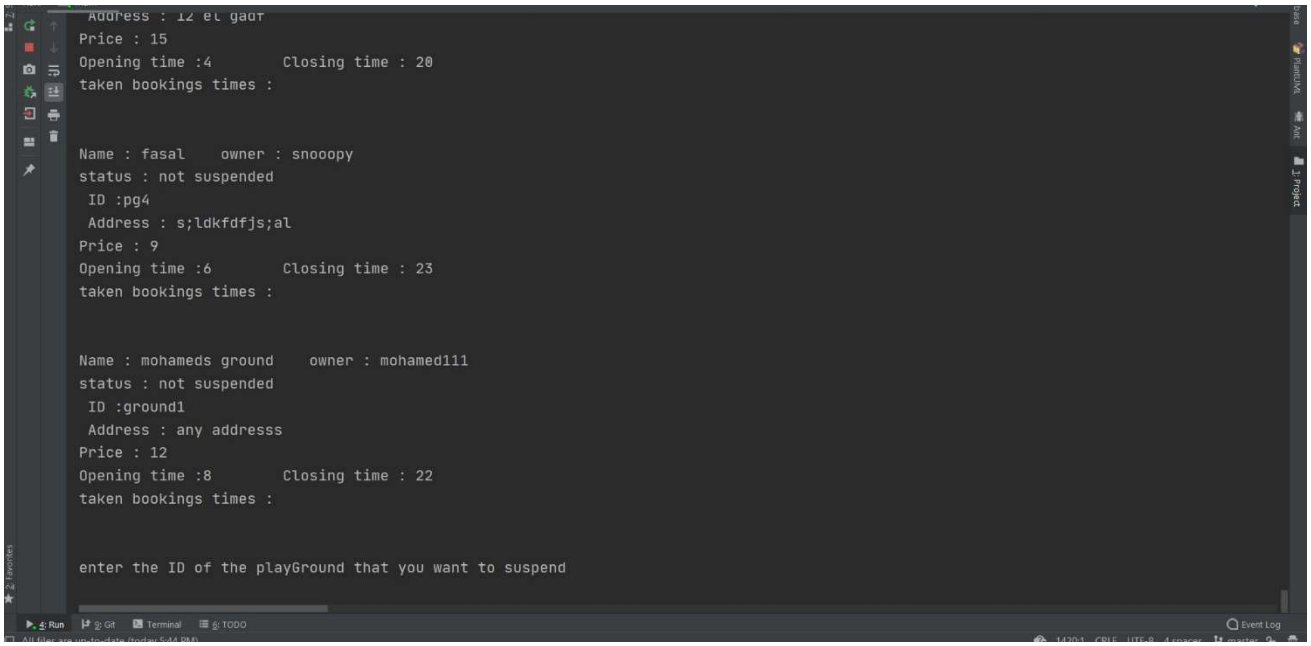
50-and now this the time to go and see the admin in this program to log in as admin all you have to do is to type admin admin in the log in form and here we go this is the admin profile you can suspend playgrounds or active or maybe delete a playground.

51-now as we see all the playgrounds are not suspended and here the list of all of them

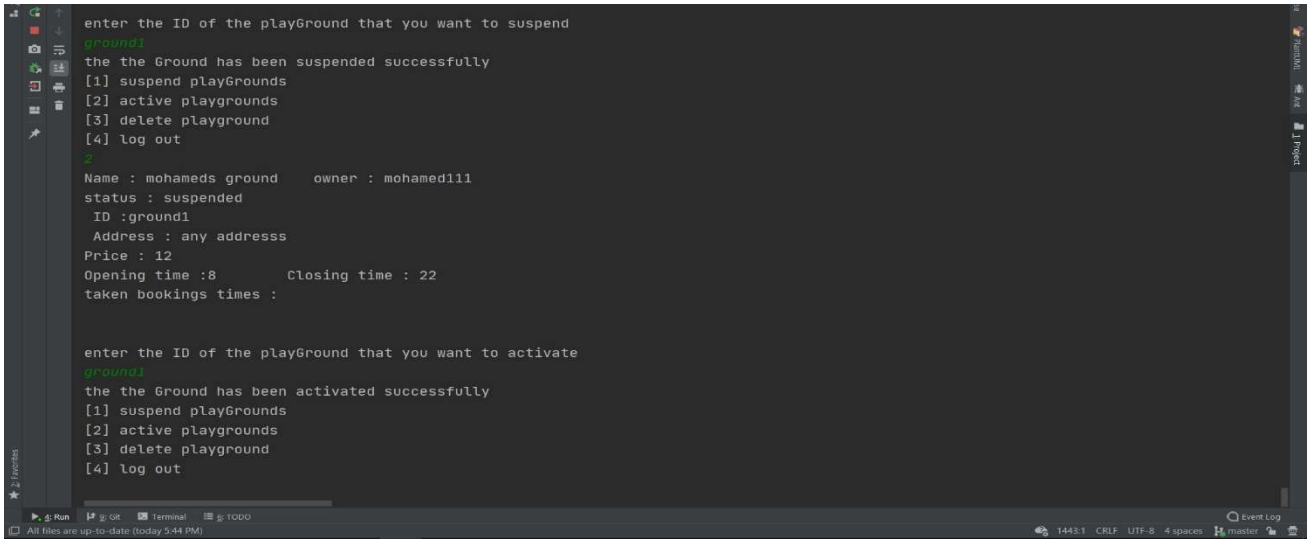


Software Design Specification

52-and this is the rest of the list of the playground and the program asking us we want to suspend a playground or not.

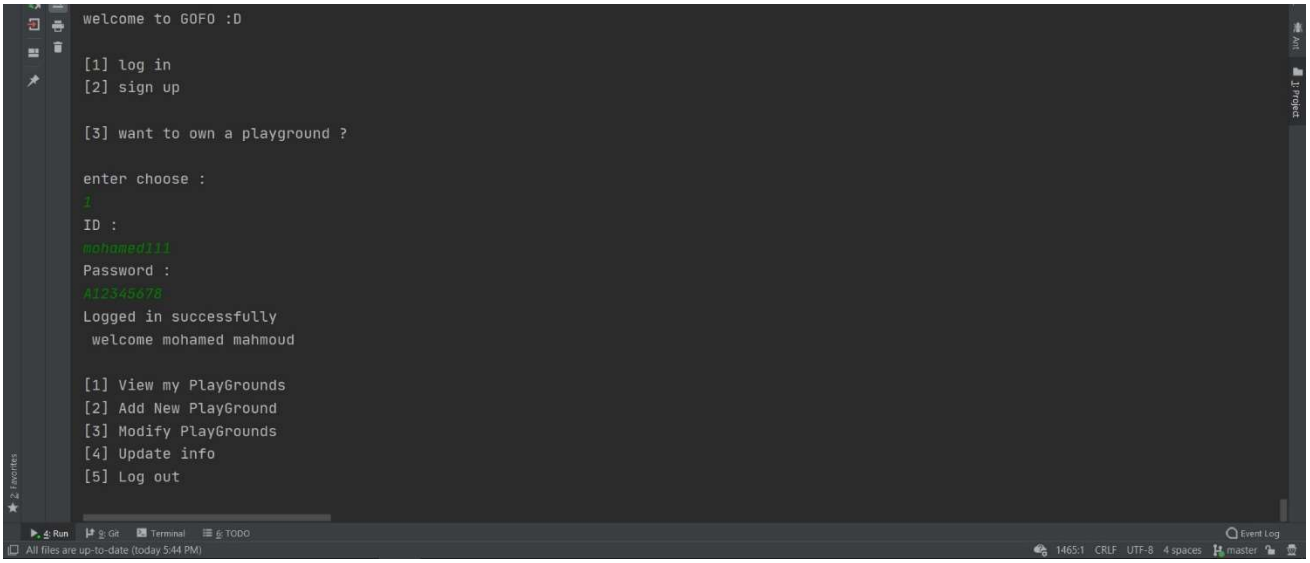


53- and as we see here we suspend ground1 which the ground we have just create and when we enter active playground will see that it is the only one suspended and we can so easily active it again by entering its ID as we see here.



Software Design Specification

54-and now I have suspended the playground ground again and lets log out and log in again to our Owner profile that we have just created and this would be good to test logging in for Owner and as we see we log in in the same place the player log in too and the app will know which account has logged in and open his profile by his account type.



55- and as we see here when we get to view our ground the status has changed and now it is suspended

Software Design Specification

The source code :

DataSource class :

```
package GOFO.DataModel;

import GOFO.OwnerClasses.Booking;
import GOFO.OwnerClasses.Ground;
import GOFO.PlayerClasses.Invitation;
import GOFO.PlayerClasses.Team;
import GOFO.Registering.I_LogIn;
import GOFO.Users.Admin;
import GOFO.Users.Owner;
import GOFO.Users.Player;
import GOFO.Users.User;

import java.util.ArrayList;
import java.util.List;

/**

 *  class is singleton class which means it has only one object
 * this class is responsible for storing all the data about the player or the play ground owner
and the admin
 * it will all the methods needed to get the data needed for the application
 * there will be in there some hard coded player and playGrounds for testing
 * this data will not be connect to any databases
 * maybe it will be stored in text file but that is not for sure yet
 *  @author mohamed
 *  * @version v1.0
 * */
public class DataSource implements I_LogIn {
    private static final DataSource dataSource= new DataSource();
     private final Admin admin ;
    private User loggedInUser;
    private final Team teamSelected;
    private final List<Player> players;
    private final List<Owner> owners;
    private final List<User> users;
    private final List<Team> teams;
    private final List <Invitation> invitations;

    /**
     * it return the static variable for the singleton class
     * @return dataSource - which the a static variable that will be used all over the program
     */
    public static DataSource getInstance() {

        return dataSource;
    }


    private DataSource (){
        teamSelected=null;
```

Software Design Specification

```
        teams = new ArrayList<Team>();
        users= new ArrayList<User>();
        admin = new Admin("admin","admin","admin@gmail.com","admin");
        invitations = new ArrayList<Invitation>();
        players = new ArrayList<Player>();
        owners = new ArrayList<Owner>();


    }

    //  ////////////////////users
    /*

     */

    /**
     * checks the ID of the user whether a player or an owner and if it exist it return false
so it is not valid and
     * duplicated and cant be signed
     * @param ID the ID of the player of the Owner
     * @return true if the the ID is valid means it is not exist before in the list and it has
an unique value
     * and false if it was int the list of users
     */
    public  boolean  checkUserIDIfValid(String ID){
        for(User i: users){
            if(ID.equals(i.getID())){
                return false;
            }
        }
        return true;


    }

    /**
     * it adds a new user to the list of users and he can be a player or an owner
     * @param newUser a new user who has just signed up
     */
    public void addNewUser(User newUser){
        users.add(newUser);
    }

    /**
     * prints all the users that is in the system form Player and Owners
     */
    public void printUsers(){
        for(User i :users){
            System.out.println(i);
        }
    }

    /**
     * it add a lot of hard coded data form player and owner and playground to test the program
on
     * because there is no database to store the data in in the current version
     */
```

Software Design Specification

```
    public void hardCodedUsers(){
        Player player1 = new Player("mohamed","mohamed ali","jerry@gamil.com ","Mohamed11");
        player1.addBookingID("pk11");
        player1.addBookingID("pk22");
        Player player2 = new Player("snoopy22","mohamed mahmoud","jerry@gamil.com ","Jery22");
        player2.addBookingID("pk33");
        player2.addBookingID("pk44");

        Player player3 = new Player("free_for_all11","Mustafa hatem","jerry ","R12345678");

        Player player4 = new Player("fresll","mohamed mahmoud","jerry ","W12345678");
        Player player5 = new Player("whatthehack","mohamed mahmoud","jerry ","E12345678");
        Player player6 = new Player("gamer","mohamed ali","jerry ","T12345678");
        Player player7 = new Player("snower","mohamed mahmoud","jerry ","Y12345678");

        Player player8 = new Player("free_for_all","Mustafa hatem","jerry ","U12345678");

        Player player9 = new Player("killer","mohamed mahmoud","jerry ","X12345678");
        Player player10 = new Player("jerry33","mohamed mahmoud","jerry ","Z12345678");
        Owner owner1 = new Owner("owner4","roma","jerry@gamil.conm","V12345678");
//        Ground ground1= new Ground()

        Owner owner2 = new Owner("owner3","roma","jerry@gamil.con","Asa7be.com");
        Ground ground1 = new Ground("pg1","romaspalyground","aiaa
jalsdkfj;aslkdfjajl;","sdfsdf"
                ,owner2.getID(),10,22,10,false);
        Booking booking1 = new Booking(10,12,"pk11","jerry11");

        Booking booking2 = new Booking(12,13,"pk22","jerry11");

        Booking booking3 = new Booking(16,17,"pk33","snoopy22");

        Booking booking4 = new Booking(18,20,"pk44","snoopy22");
        ground1.addBooking(booking1);
        ground1.addBooking(booking2);

        ground1.addBooking(booking3);

        ground1.addBooking(booking4);









        Ground ground2 =  new Ground ("pg2","comeon","lets play
","s;ldkfjs;al",owner2.getID(),8,22,10,false);
        owner2.addNewPlayGround(ground1);
        owner2.addNewPlayGround(ground2);
        Owner owner3 = new Owner("snooopy","roma","jerry@gamil.con","Asa7be,com");
        Ground ground3 = new Ground("pg3","snooopys play goutnd","lests go;","12 el gadf"
                ,owner3.getID(),4,20,15,false);
        Ground ground4 =  new Ground ("pg4","fasal","yea baby
","s;ldkfdfjs;al",owner3.getID(),6,23,9,false);
        owner3.addNewPlayGround(ground3);
        owner3.addNewPlayGround(ground4);
        Owner owner4 = new Owner("owner1","roma","jerry@gamil.con","AS123456");
```

Software Design Specification

```
        Owner owner5 = new Owner("onwer2","roma","jerry@gamil.con","MM12345678");
        users.add(player1);users.add(player2);users.add(player3);

users.add(player4);users.add(player5);users.add(player6);users.add(player7);users.add(player8);
        users.add(player9);users.add(player10);users.add(owner1);
        users.add(owner2);users.add(owner3);users.add(owner4);users.add(owner5);
        for(User i : users){
            if(i.getType().equals("Player")){
                players.add((Player) i);
            }else if(i.getType().equals("Owner")){
                owners.add((Owner) i);
            }
        }


    }

    /**
     * while the player or the owner are logging in this method checks if the given ID and
password are correct
     * or not
     * @param ID the user ID that he uses to log in
     * @param password the user password that he uses to log in
     * @return return return true if the data given were correct and false if it was wrong
     */
    @Override
    public boolean checkInfo(String ID, String password) {
        if(ID.equals("admin") && password.equals("admin")){
            loggedInUser =  admin;
            return true;
        }
        for(User i:users){
            if(i.getID().equals(ID)){
                if(i.getPassWord().equals(password)){
                    loggedInUser = i;
                    return true;
                }
            }
        }
        return false;
    }

    /**
     * it returns the User who has just logged in successfully
     * @return a User who has just logged in
     */
    @Override
    public User logIn() {
        return loggedInUser;
    }

    /**
     * return a list of all the user of the system
     * @return a list of all the users
     */
    public List<User>getUsers(){
        return users;
```

Software Design Specification

```
    }



    ////////////////////////////// Player

    /**
     * checks if the given ID of the player is exist in the Users list or not
     * @param ID the Player ID
     * @return boolean when the Player is found to check if it is exist or not and this
different from
     * is valid or not in case of valid we check if it was there wer return false not true
     */
    public boolean checkPlayerIDIfExist(String ID){
        for(User i: users){
            if(ID.equals(i.getID())){
                if(i.getType().equals("Player"))
                    return true;
            }
        }
        return false;

    }

    /**
     * this method takes the player ID and search  throw all the invitations and gives us a
list of invitations '
     * that sent to that player
     * @param ID the player ID
     * @return ArrayLIst of Invitations

     */
    public ArrayList<Invitation> getMyReceivedInvitations(String ID){
        ArrayList<Invitation> receivedInvitations = new ArrayList<Invitation>();
        for(Invitation i: invitations){
            for(String w : i.getReceiversID() ){
                if(w.equals(ID))
                    receivedInvitations.add(i);
            }
        }
        return receivedInvitations;

    }

    /**
     * it adds a new player to the players list that contain all the player in the system
     * @param newPlayer a brand new player who has just signed up
     */
    public void addNewPlayer(Player newPlayer){
        players.add(newPlayer);
    }


    /**
     * return a list of the player in the system
     *
     */
```

Software Design Specification

```
    public List<Player> getPlayers(){
        return players;
    }

    /**
     * yoy give this function the IDs of the bookings that the player has booked and it return
the
     * actual bookings
     * @param bookedIDs the IDs of the bookings that the player has booked before
     * @return a list of bookings class that the player has booked
     */
    public List<Booking>getPlayerBookings(List<String> bookedIDs){
        List<Booking> PlayerBookings= new ArrayList<Booking>();
        for(Owner i : owners){
            for(Ground w : i.getPlaygrounds()){
                for(Booking q : w.getBookings()){
                    for(String e : bookedIDs){
                        if(e.equals(q.getID()))
                            PlayerBookings.add(q);
                    }
                }
            }
        }
        return PlayerBookings;
    }



    ////////////////////////////////////////// invitations

    /**
     * it add a new invitation to the invitations list
     * @param newInvitation a brnad new invitation that will be saved in the list
     */
    public void addNewInvitation(Invitation newInvitation){
        invitations.add(newInvitation);
    }

    /**
     * it remove a the player form the sent invitation we he reject it
     * @param invID the ID of the invitation
     * @param iD the ID of the player
     */
    public void deleteReceiverByItsID( String invID, String iD){
        for(Invitation i :invitations){
            if(i.getID().equals(invID)){
                i.getReceiversID().remove(iD);
                }

            }
        }


    ////////////////// teams

    /**
```

Software Design Specification

```
     * adds a brand new team to the list of teams
     * @param newTeam a brand new team
     */
    public void addNewTeam(Team newTeam){
        teams.add(newTeam);

    }

    /**
     * display all of the team in the teams list
     */
    public void printTeams(){
        for(Team i :teams){
           i.display();
        }
    }

    /**
     * it return the team by its leader ID
     * @param PlayerID the team Leader ID
     * @return return the team of the ID were correct and return null of false
     */
    public Team getTeamByPlayerID(String PlayerID){
        for(Team i: teams){
            if(i.getTeamLeaderID().equals(PlayerID))
                return i;
        }
        return null;
    }

    /**
     * return the team by the team ID
     * @param iD the team ID
     * @return if  the ID is correct it will return the team if not will return null
     */
    public Team getTeamByID(String iD){
        for(Team i: teams){
            if(i.getID().equals(iD))
                return i;
        }
        return null;
    }

    /**
     * if the Team ID were already exist in the list of the team in will return false
     * if it is a unique one it will return true
     * @param ID the team ID
     *
     */
    public boolean checkTeamIDIfValid(String ID){
        for( Team i: teams){
            if(i.getID().equals(ID))
                return false;
        }
        return  true;
    }
```

Software Design Specification

```
    //////////////Owner

    /**
     * checks if the given ID of the Owner is exist in the Users list or not
     * @param ID the Player ID
     * @return boolean when the Player is found to check if it is exist or not and this
different from
     * is valid or not in case of valid we check if it was there wer return false not true
     */
    public boolean checkPlayGroundIDIfValid(String ID){
        for(User i: users){
            if(i.getType().equals("Owner")){
                Owner w = (Owner) i;
                for(Ground q : w.getPlaygrounds()){
                    if(q.getID().equals(ID)){
                        return false;
                    }

                }

            }
        }
        return true;
    }

    /**
     * add a a brand new Owner to the list of Owners
     * @param newOwner a new Owner who has just signed up
     */
    public void addNewOwner(Owner newOwner){
        owners.add(newOwner);
    }

    /**
     * return all the owner in the system in a list
     * @return list of all the Owners
     */
    public  List<Owner> getOwners(){
        return owners;
    }
}
```

Booking class :

```
package GOFO.OwnerClasses;

/**
 * this class contain all the info needed about the booking and it only exist int he a list of
every play ground
 *
 * @author Abdelraham fasel
 * @version v1.0
 *
 *
```

Software Design Specification

```
 */
public class Booking {
   private static int NumID;
    private String ID;
    private int startingDate;
    private  int endingDate;
    private int duration;
    private String playerID;
    private String ownerName;
    private String ownerID;
    private String playgroundName;
    private String playGroundID;
    public Booking(int startingDate, int endingDate,String ID , String playerID) {
        this(startingDate,endingDate,-1,playerID,"none","none","none","none");
        this.ID = ID;


    }
    public Booking(int startingDate, int endingDate) {
        this(startingDate,endingDate,-1,"none","none","none","none","none");
        ID = "bk"+ NumID++;

    }

    public Booking() {
        this(-1,-1,-1,"none","none","none","none","none");
        ID = "bk"+ NumID++;
        duration= endingDate- startingDate;


    }

    public Booking(int startingDate, int endingDate, int duration, String playerID, String
ownerName, String ownerID, String playgroundName, String playGroundID) {
        this.startingDate = startingDate;
        this.endingDate = endingDate;
        this.duration = duration;
        this.playerID = playerID;
        this.ownerName = ownerName;
        this.ownerID = ownerID;
        this.playgroundName = playgroundName;
        this.playGroundID = playGroundID;
        ID = "bk"+ NumID++;

    }

    public Booking(int startingDate, int endingDate, int duration, String playerID, String
ownerID) {
        this.startingDate = startingDate;
        this.endingDate = endingDate;
        this.duration = duration;
        this.playerID = playerID;
        this.ownerID = ownerID;
        ID = "bk"+ NumID++;

    }
```

Software Design Specification

```
public int getStartingDate() {
    return startingDate;
}

public void setStartingDate(int startingDate) {
    this.startingDate = startingDate;
}

public int getEndingDate() {
    return endingDate;
}

public void setEndingDate(int endingDate) {
    this.endingDate = endingDate;
}

public int getDuration() {
    return duration;
}

public void setDuration() {
    duration = endingDate-startingDate;
}

public String getPlayerID() {
    return playerID;
}

public void setPlayerID(String playerID) {
    this.playerID = playerID;
}

public String getOwnerID() {
    return ownerID;
}

public void setOwnerID(String ownerID) {
    this.ownerID = ownerID;
}

public static int getNumID() {
    return NumID;
}

public static void setNumID(int numID) {
    NumID = numID;
}

public String getID() {
    return ID;
}

public void setID(String ID) {
    this.ID = ID;
}
```

Software Design Specification

```
    public String getOwnerName() {
        return ownerName;
    }

    public void setOwnerName(String ownerName) {
        this.ownerName = ownerName;
    }

    public String getPlaygroundName() {
        return playgroundName;
    }

    public void setPlaygroundName(String playgroundName) {
        this.playgroundName = playgroundName;
    }

    public String getPlayGroundID() {
        return playGroundID;
    }

    public void setPlayGroundID(String playGroundID) {
        this.playGroundID = playGroundID;
    }

    public  void displayForGround(){
        System.out.println("ID " +ID +" staring time : "+ startingDate +"ending time : "
+endingDate);

    }
    public  void displayForPlayer(){
        System.out.println("booking ID : "+ ID);
        System.out.println("PlayGround name is " + playgroundName +" ID :"+playGroundID +"your
ID : " +playerID);
        System.out.println("The owner of the Playground ID :" + ownerID);
        System.out.println("booking stars at : " + startingDate);
        System.out.println("booking ends at : "+ endingDate);
        System.out.println("Duration :" +duration);


    }
}
```

Ground class :

```
package GOFO.OwnerClasses;

import GOFO.DataModel.DataSource;

import java.util.ArrayList;
import java.util.List;

import static java.lang.Character.*;

/**
 * this class has all the info needed about any playground and it is only exist in the Owner
```

Software Design Specification

```
class
 * @author mohamed mahmoud
 * @version 1.0
 */
public class Ground {

    private String ID;
    private String name;
    private String description;
    private String address;
    private String OwnerID;
    private int openingTime;
    private int closingTime;
    private int price ;
    private List<Booking> bookings;
    private boolean isSuspended;

    public Ground(String ID, String name, String description, String address, int openingTime,
int closingTime, int price, List<Booking> bookings, boolean isSuspended) {
        this.ID = ID;
        this.name = name;
        this.description = description;
        this.address = address;
        this.openingTime = openingTime;
        this.closingTime = closingTime;
        price = price;
        this.bookings = bookings;
        this.isSuspended = isSuspended;
    }

    public Ground(String ID, String name, String description, String address, int price,
List<Booking> bookings, boolean isSuspended) {
        this.ID = ID;
        this.name = name;
        this.description = description;
        this.address = address;
        price = price;
        this.bookings = bookings;
        this.isSuspended = isSuspended;
    }

    public Ground(String ID, String name, String description, String address, String ownerID,
int openingTime, int closingTime, int price, boolean isSuspended) {
        this.ID = ID;
        this.name = name;
        this.description = description;
        this.address = address;
        OwnerID = ownerID;
        this.openingTime = openingTime;
        this.closingTime = closingTime;
        this.price = price;
        this.isSuspended = isSuspended;
        bookings = new ArrayList<Booking>();
    }
    public Ground(){
        this.ID = "none";
        this.name ="none";
```

Software Design Specification

```
        this.description = "none";
        this.address = "none";
        this.OwnerID = "none";
        price = 0;
        this.bookings = new ArrayList<Booking>();
        this.isSuspended = false;
        this.openingTime = -1;
        this.closingTime = -1;

    }



    public String getID() {
        return ID;
    }

    public void setID(String ID) {
        this.ID = ID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public List<Booking> getBookings() {
        return bookings;
    }
```

Software Design Specification

```
    public void setBookings(List<Booking> bookings) {
        this.bookings = bookings;
    }

    public boolean isSuspended() {
        return isSuspended;
    }

    public void setSuspended(boolean suspended) {
        isSuspended = suspended;
    }

    public String getOwnerID() {
        return OwnerID;
    }

    public void setOwnerID(String ownerID) {
        OwnerID = ownerID;
    }

    public int getOpeningTime() {
        return openingTime;
    }

    public void setOpeningTime(int openingTime) {
        this.openingTime = openingTime;
    }

    public int getClosingTime() {
        return closingTime;
    }

    public void setClosingTime(int closingTime) {
        this.closingTime = closingTime;
    }

    /**
     * checks if the ID exist in the date source or not and if it there so it is not valid and
return false
     * and if it not there will return true
     * @param ID the ID of the playGround
     * @return true of the ID is valid and does not exist in the data source and false if the
opposite
     */
    public boolean signUpID(String ID){
        if( DataSource.getInstance().checkPlayGroundIDIfValid(ID)) {
            for (int i = 0; i < ID.length(); i++) {
                char a = ID.charAt(i);
                if (!isDefined(a) || isWhitespace(a)) {
                    return false;
                }
            }
        }else {
            return false;
        }
        this.ID = ID;
        return true;
```

Software Design Specification

```
        }

    /**
     * check to see if the name is valid or not
     * @param name The name entered by the user
     * @return true if the name is correct
     */
    public boolean signUpName(String name){
        name = name.trim();
        for(int i=0;i<name.length();i++){
            char a = name.charAt(i);
            if(!isLetter(a) &&!isWhitespace(a)){
                return false;
            }

        }
        this.name = name;
        return true;
     }

    /**
     * checks if the opening time is in the valid range between 0 and 24
     * @param opening the opening time entered by the Owner
     * @return True of the opening time is valid
     */
     public  boolean signUpOpeningTime(int opening){
        if(opening>=0 &&opening <24){
            this.openingTime= opening;
            return true;
        }
        return false;
     }
    /**
     * checks if the closing time is in the valid range between 0 and 24
     * @param closing the closing time entered by the Owner
     * @return True of the closing time is valid
     */
     public boolean signUpClosingTime(int closing){
        if(closing>=0 &&closing <24){
            if(closing>openingTime){
                closingTime = closing;
                return true;
            }
        }
        return false;
     }

    /**
     * check is the price is valid or not it should be more than 0 and less than 20$
     * @param price the price entered by the Owner
     * @return return true of the price is valid and false if not
     */
    public boolean signUpPrice(int price) {
        if(price >0 && price <= 20){
            this.price = price;
            return  true;
```

Software Design Specification

```
        }
        return false;
    }

    /**
     * this method checks if the given start time and the end time is valid that it is not
taken or out
     * of the starting and ending time and if all the giving info is true it will create a
Booking object
     * and add to list of bookings
     * @param start the starting time entered by the Player
     * @param end the ending time entered by the Player
     * @param playerID the ID of the player that is going to booking this playground
     * @return return true of the gavin info is true and false if not
     */
    public boolean book(int start ,int end,String playerID){
       boolean s =true ;
       boolean n =true;
       int d = end -start;
               if(!(start>=0 &&start <24))
                   return false;
            if(!(end>=0 &&end <24))
                return false;
              if(end<=start)
                  return false;


               for(Booking i:bookings){
                   for(int w= start ;w<=end;w++){
                      if(i.getStartingDate()==w){
                          s= i.getStartingDate() == end;
                      }
                      if(i.getEndingDate()==w){
                          n =i.getEndingDate()==start;
                      }

                   }


       }
               if(s &&n){
                   Booking newBooking = new Booking();
                   newBooking.setStartingDate(start);
                   newBooking.setEndingDate(end);
                   newBooking.setDuration();
                   newBooking.setPlayerID(playerID);
                   newBooking.setOwnerID(OwnerID);
                   newBooking.setPlayGroundID(ID);
                   newBooking.setPlaygroundName(name);
                   bookings.add(newBooking);

               }



    return s &&n;
    }
```

Software Design Specification

```
    /**
     * displays all the info need for the playground and all of its bookings
     */
    public void display(){

         System.out.println("Name : " + name +"    owner : " +OwnerID);
         System.out.println("status : " +(isSuspended ?"suspended" :"not suspended"));
         System.out.println(" ID :" + ID);
         System.out.println(" Address : "+ address);
         System.out.println("Price : " + price);
         System.out.println("Opening time :" +openingTime+"        Closing time : "+
closingTime);

         System.out.println("taken bookings times : ");
         for(Booking i : bookings){
             i.displayForGround();
         }
     }

    /**
     * add a brand new booking to the bookings list
     * @param newBooking the new create book
     */
     public  void addBooking(Booking newBooking){
        bookings.add(newBooking);
     }

    /**
     * gets the last booking ID
     * @return a string of the last booking ID
     */
     public String getLastBookingID(){
        return bookings.get(bookings.size()-1).getID();

     }
    /**
     * this method checks if the given start time and the end time is valid that it is not
taken or out
     * of the starting and ending time
     * @param start the starting time entered by the Player
     * @param end the ending time entered by the Player
     * @return return true of the gavin info is true and false if not
     */
     public boolean checkBooking(int start , int end){
         boolean s =true ;
         boolean n =true;
         int d = end -start;
         if(!(start>=0 &&start <24))
             return false;
         if(!(end>=0 &&end <24))
             return false;
         if(end<=start)
             return false;

         for(Booking i:bookings){
             for(int w= start ;w<=end;w++){
```

Software Design Specification

```
                if(i.getStartingDate()==w){
                    s= i.getStartingDate() == end;
                }
                if(i.getEndingDate()==w){
                    n =i.getEndingDate()==start;
                }

            }


        }
        return s &&n;

    }

}
```

User class :

```
package GOFO.Users;
/**
 * this is the base class for the users that the basic info
 * need
 * @author abderam fesal
 * @version v1.0


 */
public class User {
   protected String ID;
    protected String name;
   protected String Email;
    protected String passWord;
    protected String type;

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public User() {
        this("none","none","none","none");
        this.type= "none";

    }
    public User(String ID, String name, String email, String passWord) {
        this.ID = ID;
        this.name = name;
        Email = email;
        this.passWord = passWord;
    }
    public String getID() {
```

Software Design Specification

```
        return ID;
    }

    public void setID(String ID) {
        this.ID = ID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return Email;
    }

    public void setEmail(String email) {
        Email = email;
    }

    public String getPassWord() {
        return passWord;
    }

    public void setPassWord(String passWord) {
        passWord = passWord;
    }


}
```

Player class :

```
package GOFO.Users ;

import GOFO.DataModel.DataSource;
import GOFO.OwnerClasses.Booking;
import GOFO.PlayerClasses.Invitation;
import GOFO.PlayerClasses.Team;
import GOFO.Registering.I_SignUp;
import GOFO.Registering.I_UdataInfo;

import java.util.ArrayList;
import java.util.List;

import static java.lang.Character.*;
/**
 *class that contain every thing about the Player and his bookings
 * @author mohamed mahomud

 */
```

Software Design Specification

```
public class Player extends User implements  I_SignUp , I_UdataInfo {
    private String myTeamID;
    private List<String> joinedTeamsIDs;
    private List <String> bookingsID;

    public List<String> getBookingsID() {
        return bookingsID;
    }

    public void setBookingsID(List<String> bookingsID) {
        this.bookingsID = bookingsID;
    }

    //constructors
    public Player(){
        super();
        type = "Player";
        joinedTeamsIDs = new ArrayList<String>();
        myTeamID="none";
        bookingsID= new ArrayList<String>();


    }

    public Player(String ID, String name, String email, String passWord) {
        super(ID, name, email, passWord);
        type = "Player";
        joinedTeamsIDs = new ArrayList<String>();
        myTeamID="none";
        bookingsID= new ArrayList<String>();



    }



// sgining up

    /**
     * this method is making sure that the name the user enters is vaild
     * is only valid is it is english char only
     * and return true when its right and false when the name is wrong
      @author mohamed mohmoud said
     * @param name the given name by the user
     * @return true if the name were valid
     */
    @Override
    public boolean signUp_name(String name) {
        name = name.trim();
        for(int i=0;i<name.length();i++){
            char a = name.charAt(i);
            if(!isLetter(a) &&!isWhitespace(a)){
                return false;
            }

        }
```

Software Design Specification

```
        this.name = name;
        return true;
    }

    /**
     *  it checks if the given ID were valid or not and if it is taken before or not
     * @author mohamed mahmoud
     * @param ID the given ID by the user
     * @return true if the ID is valid false if not

     *
     */
    @Override
    public boolean signUp_ID(String ID) {
        if( DataSource.getInstance().checkUserIDIfValid(ID)) {
            for (int i = 0; i < ID.length(); i++) {
                char a = ID.charAt(i);
                if (!isDefined(a) || isWhitespace(a)) {
                    return false;
                }
            }
        }else {
            return false;
        }
        this.ID = ID;
        return true;
    }

    /**
     * checks if the the email entered is valid or not from its string only
     * @param Email the email entered by the user
     * @return true if the email were valid
     */
    @Override
    public boolean signUp_Email(String Email) {
        String regex = "^[\\w-_\\.+]*[\\w-_\\.]\\@([\\w]+\\.)+[\\w]+[\\w]$";
        this.Email = Email;
        return Email.matches(regex);


    }
    /**
     * checks if the password is valid or not it is valid if it has at least one capital
alphabet and one number and
     * at least 8 char
     * @param password the password entered by the Owner
     * @return true if the password is valid false if not
     */
    @Override
    public boolean signUp_password(String password) {
        boolean num_check = false;
        boolean cap_check = false;
        if(password.length()<8)
            return false;
        for(int i=0 ;i<password.length();i++){
            if(isUpperCase(password.charAt(i)))
                cap_check = true;
```

Software Design Specification

```
            if(isDigit(password.charAt(i)))
                num_check=true;

        }
        this.passWord = password;
        return num_check && cap_check;
    }
    /**
     * creates a new account by adding it to dataSource class
     */
    @Override
    public void create_account() {
        DataSource.getInstance().addNewUser(this);
        DataSource.getInstance().addNewPlayer(this);

    }


    //getters

    public String getMyTeamID() {
        return myTeamID;
    }
    public List<String> getJoinedTeamsIDs() {
        return joinedTeamsIDs;
    }
    @Override
    public String getType() {
        return type;
    }
    public Team getMyTeam(){
        return  DataSource.getInstance().getTeamByPlayerID(ID);

    }


    //setters


    public void setMyTeamID(String myTeamID) {
        this.myTeamID = myTeamID;
    }



    public void setJoinedTeamsIDs(List<String> joinedTeamsIDs) {
        this.joinedTeamsIDs = joinedTeamsIDs;
    }




//update
    @Override
    public boolean chanceName(String name) {
```

Software Design Specification

```
        return false;
    }

    @Override
    public boolean chancePassword(String password) {
        return false;
    }

    //adders

    /**
     * adds a brands new team to the dataSource and this team is the team that the player owns
     * @param team the new team
     */
    public void addMyTeam(Team team){
         myTeamID = team.getID();
        DataSource.getInstance().addNewTeam(team);

    }
    /**
     * adds a brands new team to the dataSource and this team is the team that the player
joined
     * @param teamID the new team ID
     */
    public void addJoinedTeam(String teamID){
        joinedTeamsIDs.add(teamID);
    }

 // invitations

    /**
     * send a new invitation to the other players to join his team
     * @param invitation the new invitation
     */
    public void sendInvitation(Invitation invitation){
        invitation.send();
    }

    /**
     * gets all the invitations that has been sent to the player and put it in  a list
     * @return a list of all the invitation that has been sent to the player
     */
    public ArrayList<Invitation> receiveInvitations(){
        return DataSource.getInstance().getMyReceivedInvitations(ID);

    }



    @Override
    public String toString() {
        return "type "+ type +"name : " +name+ " ID :" + ID + " Email : "+ Email + " password"
+ passWord +"\n";
    }

    /**
     * adds a new booking the player by saving only the ID of the booking
```

Software Design Specification

```
     * @param bookingID the booking ID
     */
    public void addBookingID(String bookingID){
        bookingsID.add(bookingID);

    }

    public List<Booking> getMyBookings(){
        return  DataSource.getInstance().getPlayerBookings(bookingsID);


    }


}
```

Owner class :

```
package GOFO.Users;

import GOFO.DataModel.DataSource;
import GOFO.OwnerClasses.Ground;
import GOFO.Registering.I_SignUp;
import GOFO.Registering.I_UdataInfo;

import java.util.ArrayList;
import java.util.List;

import static java.lang.Character.*;

/**
 *class that contain every thing about the owner and his playerGrounds
 * @author abderaham fesal

 */
public class Owner extends User implements  I_SignUp , I_UdataInfo {
    private String accountNumber;
    private String address;
    private List<Ground>playgrounds;

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public List<Ground> getPlaygrounds() {
        return playgrounds;
    }

    public void setPlaygrounds(List<Ground> playgrounds) {
        this.playgrounds = playgrounds;
    }
```

Software Design Specification

```
    public void signUp_address(String address){
        this.address=address;
    }

    /**
     * checks of the account number is valid or not it should be digits and only 12
     * @param accountNumber the account number entered from the Owner
     * @return true if the account is valid false if not
     */
    public boolean signUp_accountNumber(String accountNumber){
        if(!(accountNumber.length()==12))
            return false;
        for (int i=0;i<accountNumber.length();i++){
            if(!isDigit(accountNumber.charAt(i))){
                return false;
            }
        }
        this.accountNumber=accountNumber;
        return true;

    }

    /**
     * checks if the name is valid or  not
     * @param name the name entered by the user
     * @return true if the name is valid
     */
    @Override
    public boolean signUp_name(String name) {
        name = name.trim();
        for(int i=0;i<name.length();i++){
            char a = name.charAt(i);
            if(!isLetter(a) &&!isWhitespace(a)){
                return false;
            }

        }
        this.name = name;
        return true;
    }

    public Owner() {
        super();
        type = "Owner";
        accountNumber= "none";
        address  = "none";
        playgrounds = new ArrayList<Ground>();


    }

    public Owner(String ID, String name, String email, String passWord) {
        super(ID, name, email, passWord);
        type = "Owner";
        accountNumber= "none";
        address  = "none";
```

Software Design Specification

```
        playgrounds = new ArrayList<Ground>();

    }
    /**
     * checks if the ID exist in the date source or not and if it there so it is not valid and
return false
     * and if it not there will return true
     * @param ID the ID of the Onwer
     * @return true of the ID is valid and does not exist in the data source and false if the
opposite
     */
    @Override
    public boolean signUp_ID(String ID) {
        if( DataSource.getInstance().checkUserIDIfValid(ID)) {
            for (int i = 0; i < ID.length(); i++) {
                char a = ID.charAt(i);
                if (!isDefined(a) || isWhitespace(a)) {
                    return false;
                }
            }
        }else {
            return false;
        }
        this.ID = ID;
        return true;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public void setAccountNumber(String accountNumber) {
        this.accountNumber = accountNumber;
    }

    /**
     * checks if the the email entered is valid or not from its string only
     * @param Email the email entered by the user
     * @return true if the email were valid
     */
    @Override
    public boolean signUp_Email(String Email) {
        String regex = "^[\\w-_\\.+]*[\\w-_\\.]\\@([\\w]+\\.)+[\\w]+[\\w]$";
        this.Email = Email;
        return Email.matches(regex);

    }

    /**
     * checks if the password is valid or not it is valid if it has at least one capital
alphabet and one number and
     * at least 8 char
     * @param password the password entered by the Owner
     * @return true if the password is valid false if not
     */
    @Override
    public boolean signUp_password(String password) {
```

Software Design Specification

```
        boolean num_check = false;
        boolean cap_check = false;
        if(password.length()<8)
            return false;
        for(int i=0 ;i<password.length();i++){
            if(isUpperCase(password.charAt(i)))
                cap_check = true;
            if(isDigit(password.charAt(i)))
                num_check=true;

        }
        this.passWord = password;
        return num_check && cap_check;
    }

    /**
     * creates a new account by adding it to dataSource class
     */
    @Override
    public void create_account() {
        DataSource.getInstance().addNewUser(this);
        DataSource.getInstance().addNewOwner(this);
    }

    @Override
    public String toString() {
        return "type "+ type +"name : " +name+ " ID :" + ID + " Email : "+ Email + " password"
+ passWord +"\n";

    }

    @Override
    public boolean chanceName(String name) {
        return false;
    }

    @Override
    public boolean chancePassword(String password) {
        return false;
    }
    /**
     * adds a new playground to the list of the playgrounds
     * @param newGround the brand new playGround added by the user
     */
    public void addNewPlayGround(Ground newGround){
        playgrounds.add(newGround);
    }
}
```

Admin class :

```
package GOFO.Users;


/**
```

Software Design Specification

```
 * class that is  representing the admin
 */
public class Admin extends User  {

    public Admin() {
        super();
        type="Admin";
    }

    public Admin(String ID, String name, String email, String passWord) {
        super(ID, name, email, passWord);
        type="Admin";
    }


}
```

Verify class :

```
package GOFO.verification;

import java.util.Properties;
import java.util.Random;
import java.util.Scanner;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

/**
 * sent the verification code and check if it is right or not and has connection to the
internet
 * and to run this class you must add the lib added with the project
 */
public class Verify {
    private static Random rand ;
    private static  int rand_int1 ;

    /**
     * sent a massage to the user to his email with a random number and only work with Gmail
     * @param email the user email
     * @param name the user name
     * @return return true if the massage sent successfully and false if not
     */
    public static boolean send_verify_code(String email,String name) {
        rand = new Random();
        rand_int1 = rand.nextInt(10000);


        String from = "mohamedpop955555@gmail.com";
        String host = "smtp.gmail.com";
```

Software Design Specification

```
        Properties properties = System.getProperties();
        properties.put("mail.smtp.host", host);
        properties.put("mail.smtp.port", "465");
        properties.put("mail.smtp.ssl.enable", "true");
        properties.put("mail.smtp.auth", "true");

        Session session = Session.getInstance(properties, new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication("mohamedpop.955555@gmail.com", "asa7be.com");
            }

        });

        session.setDebug(false);
        try {

            MimeMessage message = new MimeMessage(session);

            message.setFrom(new InternetAddress(from));


            message.addRecipient(Message.RecipientType.TO, new InternetAddress(email));


            message.setSubject(" GOFO ");
            message.setText("hi " +name+"\n\nyour verification code is "+rand_int1);

            Transport.send(message);
            return true;
        } catch (
                MessagingException mex) {
//            mex.printStackTrace();
            return false;
        }

    }

    /**
     * checks if the code sent to the User and user enters right or not
     * @param code the code intered by the user
     * @return true if the code is right
     */
    public static boolean check_validation_code(int code){
        return code == rand_int1;
    }

    }
```

invitation class :

```
package GOFO.PlayerClasses;
```

Software Design Specification

```
import GOFO.DataModel.DataSource;

import java.util.ArrayList;
import java.util.List;

/**
 * it has all the info needed about the invitation and responsible for sending and receiving
the invitations
 * @author mohamed mahmoud
 */
public class Invitation {
    private static int IDNum=0;
    private String ID;
    private String massage;
    private String senderID;
    private String teamInvitedID;
    private List<String> receiversID;

    public static int getIDNum() {
        return IDNum;
    }

    public static void setIDNum(int IDNum) {
        Invitation.IDNum = IDNum;
    }

    public String getID() {
        return ID;
    }

    public void setID(String ID) {
        this.ID = ID;
    }

    public String getTeamInvitedID() {
        return teamInvitedID;
    }

    public void setTeamInvitedID(String teamInvited) {
        this.teamInvitedID = teamInvited;
    }

    public Invitation (){
        senderID= "not known yet";
        receiversID = new ArrayList<String>();
        massage="none";
        IDNum ++;
        ID = "inv"+IDNum;
        teamInvitedID = "none";
    }

    public String getMassage() {
        return massage;
    }

    public void setMassage(String massage) {
```

Software Design Specification

```
        this.massage = massage;
    }

    public  Invitation(String senderID, List<String> receiversID, String massage){
        this.senderID = senderID;
        this.receiversID = receiversID;
        this.massage = massage;
        IDNum ++;
        ID = "inv"+IDNum;
    }

    public String getSenderID() {
        return senderID;
    }

    public void setSenderID(String senderID) {
        this.senderID = senderID;
    }

    public List<String> getReceiversID() {
        return receiversID;
    }

    public void setReceiversID(List<String> receiversID) {
        this.receiversID = receiversID;
    }

    /**
     * add the current invitation to dataSource to be assessable to all the Players
     */
    public void send(){
        DataSource.getInstance().addNewInvitation(this);
    }

    /**
     * adds a player ID to list of the receivers which is the player that is going to receive
this invitation
     * @param ID the player ID
     */
    public void addReceiver(String ID){
        receiversID.add(ID);
    }

    /**
     * check if the Player ID gavin is valid or not
     * @param ID the player ID
     * @return return true if the ID is right and it is the sender ID and false if is not valid
     */
    public boolean checkReceiverID( String ID){
        if(ID.equals(senderID))
            return false;
        for(String i: receiversID){
            if(i.equals(ID))
                return false;
        }
        return DataSource.getInstance().checkPlayerIDIfExist(ID);
```

Software Design Specification

```
    }

    /**
     * display all the info need about the invitation
     */
    public void display (){
        System.out.println("Invitation ID :" +ID);
        System.out.println(senderID+" wants you to join his teem " );
        System.out.println("content of the massage :" +massage);
        System.out.println(" Invited team ID "+teamInvitedID);
    }

    /**
     * display all the players that has received this invitation
     */
    public void displayReceivers(){
        for(String ID : receiversID){
            System.out.print(ID +"  ");
        }
    }

    /**
     * take the player ID and remove ID from the list that has all the IDs of the receivers
     * @param ID the ID of the player that is going to be deleted
     */
    public  void deleteReceiverByItsID(String ID){
        DataSource.getInstance().deleteReceiverByItsID(this.ID,ID);

    }
}
```

Team class :

```
package GOFO.PlayerClasses;

import GOFO.DataModel.DataSource;

import java.util.ArrayList;
import java.util.List;

import static java.lang.Character.isDefined;
import static java.lang.Character.isWhitespace;

/**
 * has all the info needed for the team
 * @author mohamed mahomud
 * @version  1.0
 */
public class Team {

    private String ID;
    private String teamLeaderID;
    private final List<String> invitedPlayersIDs;
```

Software Design Specification

```
    public Team(String ID, String teamLeaderID, List<String> invitedPlayersIDs) {
        this.ID = ID;
        this.teamLeaderID = teamLeaderID;
        this.invitedPlayersIDs = invitedPlayersIDs;
    }

    public Team(){
        invitedPlayersIDs = new ArrayList<String>();
        teamLeaderID = "none";
        ID= "none";
    }

    @Override
    public String toString() {
        return "Team ID : " +ID +" ";
    }

    /**
     * checks that the given ID is valid or not
     * @param ID the ID entered by the player
     * @return true of the ID is valid false if not
     */
    public boolean signUpID(String ID){
        if( DataSource.getInstance().checkTeamIDIfValid(ID)) {
            for (int i = 0; i < ID.length(); i++) {
                char a = ID.charAt(i);
                if (!isDefined(a) || isWhitespace(a)) {
                    return false;
                }
            }
        }else {
            return false;
        }
        this.ID = ID;
        return true;
    }
    public String getID() {
        return ID;
    }

    public void setID(String ID) {
        this.ID = ID;
    }




    public String getTeamLeaderID() {
        return teamLeaderID;
    }

    public List<String> getInvitedPlayersIDs() {
        return invitedPlayersIDs;
    }

    public void setTeamLeaderID(String teamLeaderID) {
```

Software Design Specification

```
        this.teamLeaderID = teamLeaderID;
    }

    /**
     * add a new player for the invited player ID list
     * @param PlayerID player ID
     */
    public void addInvitedPlayer(String PlayerID){
        invitedPlayersIDs.add(PlayerID);

    }

    /**
     * display all the info need about the team
     */
    public void display(){
        int w=1;
        System.out.println("Team ID : " + ID);
        System.out.println("caption : "+teamLeaderID);
        System.out.println("Players :");
        for(String i : invitedPlayersIDs ){
            System.out.println(  "          ["+ w+"]" + i);
            w++;
        }

    }


}
```

Search class :

```
package GOFO.PlayerClasses;

import GOFO.DataModel.DataSource;
import GOFO.OwnerClasses.Ground;
import GOFO.Users.Owner;

import java.util.ArrayList;
import java.util.List;

/**
 * Utility Class that has all the method needed to preform all kind of searching
 * @author abderham fesal
 * @version 1.0
 */
public class Search {
    /**
     * this method let the player view all the available playgrounds
     * @return a list of Grounds that has all the available Grounds for the player to view
     */
    public static List<Ground> ViewAllGrounds () {
        List<Owner> owners = DataSource.getInstance().getOwners();
        List<Ground> grounds= new ArrayList<Ground>();
```

Software Design Specification

```
        for(Owner i : owners){
            for(Ground w : i.getPlaygrounds()){
                if(!w.isSuspended()){
                    grounds.add(w);
                }
            }
        }


        return grounds;
    }

    /**
     * the player enter a specific range of staring and ending time and and will return a list
     * of the playGround that is available is this time
     * @param start the staring time
     * @param end   the ending time
     * @return a list of the available playgrounds
     */
    public static List<Ground>  searchBySpecificRangeOfDateAndTime(int start , int end) {
    List<Owner> owners = DataSource.getInstance().getOwners();
    List<Ground> grounds = new ArrayList<>();
    for (Owner i : owners) {
        for (Ground w : i.getPlaygrounds()) {
            if (w.checkBooking(start, end)) {
                if (!w.isSuspended()) {
                    grounds.add(w);
                }
            }
        }
    }

return grounds;
}
}
```

AdminProfile class :

```
package GOFO.contorllers;

import GOFO.DataModel.DataSource;
import GOFO.OwnerClasses.Ground;
import GOFO.Users.Owner;
import org.w3c.dom.ls.LSOutput;

import javax.xml.crypto.Data;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * this class is a control class and because this program is a console based program so it just
contain
 * a static methods that will be used in the main to keep the code more organized and will
contain
```

Software Design Specification

```
 * all the methods the the Admin will be able to preform .
 * @author mohamed mahmoud
 *
 *
 */
public class AdminProfile {
    /**
     *this class has a private constructor because it a utility class
     */
    private AdminProfile(){}

   public  static void displayMenu(){
       System.out.println("[1] suspend playGrounds");
       System.out.println("[2] active playgrounds");
       System.out.println("[3] delete playground");
       System.out.println("[4] log out");
   }
    /**
     *this method get all the play ground that is active and print it to the admin
     * so he can suspend any playground he wants by entering its ID
     *
     */
   public static  void suspendPlayGrounds() {
        int check=0;
       Scanner scanner = new Scanner(System.in);
       Scanner scanner1 = new Scanner(System.in);



       String string_input;
       List<Owner> owners = DataSource.getInstance().getOwners();
       List<Ground> aLLGrounds = new ArrayList<>();
       for (Owner i : owners) {
           for (Ground w : i.getPlaygrounds()) {
               if (!w.isSuspended()) {
                   w.display();
                   System.out.println("\n");
                   aLLGrounds.add(w);
               }
           }


       }
       while (true) {
           System.out.println("enter the ID of the playGround that you want to suspend");
           string_input=scanner.nextLine();

           for (Ground i : aLLGrounds) {
               if (i.getID().equals(string_input)){
                   i.setSuspended(true);
                   System.out.println("the the Ground has been suspended successfully");
                   check=1;
                   break;
               }
           }
           if(check==1){
               check=0;
```

Software Design Specification

```
                break;
            }else{
                System.out.println("you entered a wrong ID ");
            }


        }
    }/**
      *this method get all the play ground that is suspended and print it to the admin
      * so he can active any playground he wants by entering its ID
      */
    public static void activePlayGround(){
        int check=0;
        Scanner scanner = new Scanner(System.in);
        Scanner scanner1 = new Scanner(System.in);



        String string_input;
        List<Owner> owners = DataSource.getInstance().getOwners();
        List<Ground> aLLGrounds = new ArrayList<>();
        for (Owner i : owners) {
            for (Ground w : i.getPlaygrounds()) {
                if (w.isSuspended()) {
                    w.display();
                    System.out.println("\n");
                    aLLGrounds.add(w);
                }
            }


        }
        while (true) {
            System.out.println("enter the ID of the playGround that you want to activate");
            string_input = scanner.nextLine();

            for (Ground i : aLLGrounds) {
                if (i.getID().equals(string_input)) {
                    i.setSuspended(false);
                    System.out.println("the the Ground has been activated successfully");
                    check = 1;
                    break;
                }
            }
            if (check == 1) {
                check=0;
                break;
            } else {
                System.out.println("you entered a wrong ID ");
            }


        }

    }
     /**
      *this static method get all the play ground that is available and print it to the admin
```

Software Design Specification

```
     * so he can delete any playground he wants by entering its ID
     */
    public static void deletePlayGround(){
        int check=0;
        Scanner scanner = new Scanner(System.in);
        Scanner scanner1 = new Scanner(System.in);



        String string_input;
        List<Owner> owners = DataSource.getInstance().getOwners();
        for (Owner i : owners) {
            for (Ground w : i.getPlaygrounds()) {
                if (w.isSuspended()) {
                    w.display();
                    System.out.println("\n");

                }
            }


        }
        while (true) {
            System.out.println("enter the ID of the playGround that you delete (note : the play
ground has to be suspended fist to get deleted)");
            string_input = scanner.nextLine();
            Owner owner = null;
            Ground delGround = null;

            for (Owner i : owners) {
                for (Ground w : i.getPlaygrounds()) {
                    if (w.isSuspended()) {
                        if (w.getID().equals(string_input)) {
                            owner = i;
                            delGround = w;
//                            i.getPlaygrounds().remove(w);
                            System.out.println("the PlayGround has been deleted successfully ");
                            check = 1;
                            break;

                        }
                    }
                }

            }
            assert owner != null;
            List<Ground> grounds = owner.getPlaygrounds();
            grounds.remove(delGround);


            if (check == 1) {
                check = 0;
                break;
            } else {
                System.out.println("you entered a wrong ID ");
            }
```

Software Design Specification

```
        }

    }

}
```

HomeScreen class :

```
package GOFO.contorllers;

import GOFO.DataModel.DataSource;
import GOFO.Users.Owner;
import GOFO.Users.Player;
import GOFO.Users.User;
import GOFO.verification.Verify;

import java.util.Scanner;

/**
 * this class is a utility class ans also we can consider it as a control class that allow the
user to
 * sign up and log in through static methods that will be used in the main to reduce the code
there and keep
 * it more arranged
 * @author mohamed mahmoud
 */
public class HomeScreen {
    private HomeScreen(){}
    /**
     * this static method is responsible for the user to log in the player of the owner by
entering their
     * ID and password and it will use the data source to check for the ID and the password if
it is right
     * or not
     * @return User which is the the user how has already logged in it can be an admin or a
player or an owner
     *
     */
    public static User userLogIn(){
        String string_input;
        int int_input = 0;
        Scanner scanner = new Scanner(System.in);
        Scanner scanner1 = new Scanner(System.in);


            System.out.println("ID :");
            string_input = scanner.nextLine();
            String userID = string_input;
            System.out.println("Password :");
            string_input = scanner.nextLine();
            String userPassword = string_input;
            while (!DataSource.getInstance().checkInfo(userID, userPassword)) {
```

Software Design Specification

```
                System.out.println("you user name or pass word is incorrect ");
                System.out.println("ID :");
                string_input = scanner.nextLine();
                userID = string_input;
                System.out.println("Password :");
                string_input = scanner.nextLine();
                userPassword = string_input;
            }
            User loggedInUser = DataSource.getInstance().logIn();
            System.out.println("Logged in successfully");

            return loggedInUser;

    }
    /**
     *static method for the player to sign up it creates a new player and make the user enter
all the need
     * info to sign up the player account and it uses player class methods to sign up player
info in a correct way
     * and it has a connection to the internet through the  verify class to verify the player
Gmail
     * and it calls all the need methods for that
     * @return Player
     * which is the player who just already singed up
     */
    public static Player playerSigningUp(){
        String string_input;
        int int_input = 0;
        Scanner scanner = new Scanner(System.in);
        Scanner scanner1 = new Scanner(System.in);
        Player currentPlayer = new Player();
//        scanner.nextLine();//clean the buffer
        System.out.println("enter your full name ");
        string_input=scanner.nextLine();

        while(!currentPlayer.signUp_name(string_input)){
            //while get into this loop if the user entered a wrong user name
            //only char works
            System.out.println("the name entered is not valid only english char works");
            string_input=scanner.nextLine();

        }
        String playerFullName = string_input;
//                            scanner.nextLine();
        System.out.println("enter your ID it (must be unique one) :  ");
        string_input= scanner.nextLine();
        while(!currentPlayer.signUp_ID(string_input)){
            System.out.println("the ID you entered is not valid or maybe taken");
            string_input=scanner.nextLine();

        }
        System.out.println("accepted \n your ID is : " + string_input);
        String PlayerID= string_input;
//                            scanner.nextLine();
        System.out.println("enter you password :");
        string_input= scanner.nextLine();
        while(!currentPlayer.signUp_password(string_input)){
```

Software Design Specification

```
            System.out.println("the password you entered is not valid \n" +
                    "add at least one number and one capital alphabet and it is more than 8
character");
            string_input=scanner.nextLine();

        }

        while(true){
            System.out.println("enter your Gmail : ");
//                              scanner.nextLine();
            string_input= scanner.nextLine();
            while(!currentPlayer.signUp_Email(string_input)){
                System.out.println("the entered email is not valid");
                System.out.println("enter you Gmail :");
                string_input = scanner.nextLine();
            }
            System.out.println("sending");
            while(!Verify.send_verify_code(string_input,playerFullName)){
                System.out.println("something went wrong check you internet connection or try
to enter the email again");
                System.out.println("enter you Gmail :");
                string_input = scanner.nextLine();

            }
            System.out.println("a verification code is sent to your email");
            System.out.println("[1] enter the code");
            System.out.println("[2] resent ");
            int_input= scanner1.nextInt();
            if(int_input==1){
                System.out.println("code :");
                int_input = scanner1.nextInt();
                if(!Verify.check_validation_code(int_input)){
                    System.out.println("the verification code is wrong please enter the Email
again");
//                                              scanner.nextLine();
                    continue;
                }
                currentPlayer.create_account();
                System.out.println(" congratulations ! \nyou have registered successfully ");
                break;



            }else if(int_input ==2){
                continue;

            }else {
                System.out.println("enter a valid value !");
            }
        }

        return currentPlayer;

    }
    /**
            *static method for the owner to sign up it creates a new owner and make the user
enter all the need
```

Software Design Specification

```
     * info to sign up the owner account and it uses player class methods to sign up player
info in a correct way
     * and it has a connection to the internet through the  verify class to verify the Owner
Gmail
     * and it calls all the need methods for that
     * @return Onwer
     * which is the player who just already singed up
     */
    public static Owner ownerSigningUp(){
        String string_input;
        int int_input = 0;
        Scanner scanner = new Scanner(System.in);
        Scanner scanner1 = new Scanner(System.in);
        Owner currentOwner = new Owner();
//          scanner.nextLine();//clean the buffer
        System.out.println("enter your full name ");
        string_input=scanner.nextLine();

        while(!currentOwner.signUp_name(string_input)){
            //while get into this loop if the user entered a wrong user name
            //only char works
            System.out.println("the name entered is not valid only english char works");
            string_input=scanner.nextLine();

        }
        String playerFullName = string_input;
//                              scanner.nextLine();
        System.out.println("enter your ID it (must be unique one) :  ");
        string_input= scanner.nextLine();
        while(!currentOwner.signUp_ID(string_input)){
            System.out.println("the ID you entered is not valid or maybe taken");
            string_input=scanner.nextLine();

        }
        System.out.println("accepted \n your ID is : " + string_input);
        String PlayerID= string_input;
//                              scanner.nextLine();
        System.out.println("enter you password :");
        string_input= scanner.nextLine();
        while(!currentOwner.signUp_password(string_input)){
            System.out.println("the password you entered is not valid \n" +
                    "add at least one number and one capital alphabet and it is more than 8
character");
            string_input=scanner.nextLine();

        }
        System.out.println("enter you account number");
        string_input = scanner.nextLine();
        while(!currentOwner.signUp_accountNumber(string_input)){
            System.out.println("something is wrong (number must be 12 digits)");
            string_input = scanner.nextLine();
        }
        System.out.println("enter you address");
        string_input=scanner.nextLine();
        currentOwner.signUp_address(string_input);
```

Software Design Specification

```
        System.out.println("enter you PlayGround info");


        while(true){
            System.out.println("enter your Gmail : ");
//                              scanner.nextLine();
            string_input= scanner.nextLine();
            while(!currentOwner.signUp_Email(string_input)){
                System.out.println("the entered email is not valid");
                System.out.println("enter you Gmail :");
                string_input = scanner.nextLine();
            }
            System.out.println("sending");
            while(!Verify.send_verify_code(string_input,playerFullName)){
                System.out.println("something went wrong check you internet connection or try
to enter the email again");
                System.out.println("enter you Gmail :");
                string_input = scanner.nextLine();

            }
            System.out.println("a verification code is sent to your email");
            System.out.println("[1] enter the code");
            System.out.println("[2] resent ");
            int_input= scanner1.nextInt();
            if(int_input==1){
                System.out.println("code :");
                int_input = scanner1.nextInt();
                if(!Verify.check_validation_code(int_input)){
                    System.out.println("the verification code is wrong please enter the Email
again");
//                                              scanner.nextLine();
                    continue;
                }
                currentOwner.create_account();
                System.out.println(" congratulations ! \nyou have registered successfully ");
//                DataSource.getInstance().printUsers();
                break;



            }else if(int_input ==2){
                continue;

            }else {
                System.out.println("enter a valid value !");
            }
        }




        return currentOwner;

    }
```

Software Design Specification

```
    /**
     prints the first screen for the program that allow the user to lon in ofr sign up or sign
up as an Owner

    */
    public static void printMenu(){
        System.out.println("welcome to GOFO :D\n");
        System.out.println("[1] log in");
        System.out.println("[2] sign up\n");
        System.out.println("[3] want to own a playground ?\n");

        System.out.println("enter choose :");

    }
}
```

OwnerProfile :

```
package GOFO.contorllers;

import GOFO.OwnerClasses.Ground;
import GOFO.Users.Owner;

import java.util.InputMismatchException;
import java.util.Scanner;

/**
 * this class is a control class and because this program is a console based program so it just
contain
 *  * a static methods that will be used in the main to keep the code more organized and will
contain
 *  * all the methods that the owner will be able to preform in his profile page .
 * @author mohamed mahmoud
 *
 */
public class OwnerProfile {
    private OwnerProfile(){ }

    /**
     * prints the menu for the owner to choose from
     * @param currentOwner the owner who just logged in or signed up
     */
    public static void printMenu(Owner currentOwner){

        System.out.println(" welcome "+ currentOwner.getName()+"\n");

        System.out.println("[1] View my PlayGrounds");
        System.out.println("[2] Add New PlayGround");
        System.out.println("[3] Modify PlayGrounds ");
        System.out.println("[4] Update info");
        System.out.println("[5] Log out");
    }
```

Software Design Specification

```
    /**
     * this method create a new play ground and fill all of its info and add it to the list of
the playgrounds
     * of the current owner
     * @param currentOwner the owner who just logged in or signed up
     */
    public static void AddNewPlayGround(Owner currentOwner){
        String string_input;
        int int_input = 0;
        Ground newGround = new Ground();
        newGround.setOwnerID(currentOwner.getID());
        Scanner scanner = new Scanner(System.in);
        Scanner scanner1 = new Scanner(System.in);
        try {
            System.out.println("enter Playground ID : (must be a unique)");
            string_input = scanner.nextLine();
            while (!newGround.signUpID(string_input)) {
                System.out.println("the ID you entered is not valid or maybe taken");
                string_input = scanner.nextLine();

            }
            System.out.println("accepted \n your ID is : " + string_input);
            String GroundID = string_input;
            System.out.println("enter PlayGround name");
            string_input = scanner.nextLine();

            while (!newGround.signUpName(string_input)) {
                //while get into this loop if the user entered a wrong user name
                //only char works
                System.out.println("the name entered is not valid only english char works");
                string_input = scanner.nextLine();

            }
            String playerFullName = string_input;
            System.out.println("enter Playground address : ");
            string_input = scanner.nextLine();
            newGround.setAddress(string_input);

            System.out.println("enter playground description :");
            string_input = scanner.nextLine();
            newGround.setDescription(string_input);
            try {
                System.out.println("enter the opening time of the playground :");
                int_input = scanner1.nextInt();
                while (!newGround.signUpOpeningTime(int_input)){
                    System.out.println("the time you entered is not valid it must be between  (
0 to 24 )");
                   int_input = scanner.nextInt();

                }
                System.out.println("enter the closing time of the playground :");
                while (!newGround.signUpClosingTime(int_input)) {
                    System.out.println("the time you entered is not valid it must be between  (
0 to 24 ) and after the opening time");
                    int_input = scanner.nextInt();
                }
```

Software Design Specification

```
                // adding the playground to the array in the Owner
                System.out.println("enter the playground booking price :");
                int_input = scanner1.nextInt();
                while (!newGround.signUpPrice(int_input)) {
                    System.out.println("the price you entered is not right is has to be more
than 0 $ and less than  20 $ ");
                    int_input = scanner.nextInt();
                }

                currentOwner.addNewPlayGround(newGround);
                System.out.println("the Playground has added successfully");

            }catch (InputMismatchException e){
                System.out.println("you entered an a string value and it has to be integer ");
                    scanner1.nextLine();
            }

        }catch (InputMismatchException e){
            System.out.println("you entered an a string value and it has to be inter ");
        }

    }

    /**
     * allow the current owner  to view all of his playgrounds
     * @param currentOwner the owner who just logged in or signed up
     */
    public static void viewPlayGrounds(Owner currentOwner){
        String string_input;
        int int_input = 0;
        Scanner scanner = new Scanner(System.in);
        Scanner scanner1 = new Scanner(System.in);
        System.out.println("your playgrounds : ");
        for(Ground i : currentOwner.getPlaygrounds())
            i.display();
        System.out.println("\n");
        System.out.println("[0] back");
        string_input=scanner.nextLine();



    }
}
```

PlayerProfile :

```
package GOFO.contorllers;

import GOFO.DataModel.DataSource;
import GOFO.OwnerClasses.Booking;
import GOFO.OwnerClasses.Ground;
import GOFO.PlayerClasses.Invitation;
import GOFO.PlayerClasses.Search;
```

Software Design Specification

```
import GOFO.PlayerClasses.Team;
import GOFO.Users.Player;

import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.List;
import java.util.Scanner;

/**
 * this class is a control class and because this program is a console based program so it just
contain
 * a static methods that will be used in the main to keep the code more organized and will
contain
 * all the methods that the Player will be able to preform .
 * @author mohamed mahmoud
 */
public class PlayersProfile {
    private  PlayersProfile(){}

    /**
     * prints all menu for the player to choose from whatever he want to do
     * @param currentPlayer the player who recently logged in or signed up
     */
    public static void printMenu( Player currentPlayer){

        System.out.println("welcome back " + currentPlayer.getName() + "\n");
        System.out.println("[1] Manage booking");
        System.out.println("[2] Manage teams");
        System.out.println("[3] Mange invitations");
        System.out.println("[4] Update INFO");
        System.out.println("[5] log out");

    }

    /**
     * static method allow the player to interact with the program to search for the
playgrounds he wants
     * and can book any one of them and can view them
     * @param currentPlayer the player who recently logged in or signed up
     */
    public static void mangeBooking (Player currentPlayer){
        Scanner scanner = new Scanner(System.in);
        Scanner scanner1 = new Scanner(System.in);
        String string_input;
        int int_input = 0;

        while(true){
            System.out.println("[1]Search Playground ");
            System.out.println("[2] view my bookings");
            System.out.println("[3] back");
            try{
                int_input=scanner1.nextInt();
                if(int_input==1){
                    while(true) {
                        System.out.println("[1] view all grounds ");
                        System.out.println("[2] search by specific range of date and time ");
```

Software Design Specification

```
                        System.out.println("[3] search by rance price ");
                        System.out.println("[3] search by specific owner name");
                        System.out.println("[4] search by specific Playground name ");
                        System.out.println("[5] back");
                        try {
                            int_input = scanner1.nextInt();
                            if(int_input==1) {
                                while (true) {
                                    try {
                                        System.out.println("these are the available
playGrounds");
                                        List<Ground> availableGrounds =
Search.ViewAllGrounds();
                                        for (Ground i : availableGrounds) {
                                            i.display();
                                            System.out.println("\n\n");
                                        }
                                        System.out.println("do want to book a play ground ?
(y/n)");
                                        string_input = scanner.nextLine();
                                        if (string_input.equals("y") ||
string_input.equals("Y")) {
                                            System.out.println("enter the ID of the play ground
");
                                            string_input = scanner.nextLine();
                                            for (Ground i : availableGrounds) {
                                                if (i.getID().equals(string_input)) {
                                                    System.out.println("what time do want to
start playing at in  24 hours format ");
                                                    int_input = scanner1.nextInt();
                                                    int startingTime= int_input;
                                                    System.out.println("hat time do want to end
the match at in  24 hours format");
                                                    int_input = scanner1.nextInt();
                                                    int endingTime = int_input;
                                                    while
(!i.book(startingTime,endingTime,currentPlayer.getID())) {
                                                        System.out.println("the time you
entered is wrong or taken try again");
                                                        System.out.println("what time do want
to start playing at in  24 hours format ");
                                                        startingTime= scanner1.nextInt();
                                                        System.out.println("hat time do want to
end the match at in  24 hours format");
                                                        endingTime= scanner1.nextInt();
                                                    }
                                                    System.out.println("you has booked the
playground successfully");
currentPlayer.addBookingID(i.getLastBookingID());
                                                    break;
                                                }
                                            break;
                                        }
```

Software Design Specification

```
                                    } else
                                        break;
                                }catch (InputMismatchException e){
                                    scanner.nextLine();
                                    scanner1.nextLine();
                                    System.out.println("please enter an integer value");
                                }
                            }

                        }else if(int_input==2){
                            while(true) {
                                try {
                                    System.out.println("enter the starting time of the date
you want to search with :");
                                    int start = scanner1.nextInt();
                                    System.out.println("enter the ending time");
                                    int end = scanner1.nextInt();
                                    while (end <= start) {
                                        System.out.println("the input you just gave is not
right try again");

                                        System.out.println("enter the starting time of the
date you want to search with :");
                                        start = scanner1.nextInt();
                                        System.out.println("enter the ending time");
                                        end = scanner1.nextInt();


                                    }
                                    List<Ground> grounds =
Search.searchBySpecificRangeOfDateAndTime(start, end);
                                    if (!grounds.isEmpty()) {
                                        System.out.println("the aviliable play grounds in
this time are :");

                                        for (Ground i : grounds) {
                                            i.display();
                                            System.out.println("\n");
                                        }

                                    } else {
                                        System.out.println("there is not availible
playgrounds int the given time ");

                                        System.out.println("[0] back");
                                        string_input = scanner.nextLine();
                                        break;
                                    }
                                    System.out.println("do want to book a play ground ?
(y/n)");

                                    string_input = scanner.nextLine();
                                    if (string_input.equals("y") ||
string_input.equals("Y")) {

                                        System.out.println("enter the ID of the play ground
");

                                        string_input = scanner.nextLine();
                                        for (Ground i : grounds) {
                                            if (i.getID().equals(string_input)) {
                                                System.out.println("what time do want to
```

Software Design Specification

```
start playing at in  24 hours format ");
                                                    int_input = scanner1.nextInt();
                                                    int startingTime= int_input;
                                                    System.out.println("hat time do want to end
the match at in  24 hours format");
                                                    int_input = scanner1.nextInt();
                                                    int endingTime = int_input;

                                                    while
(!i.book(startingTime,endingTime,currentPlayer.getID())) {
                                                        System.out.println("the time you
entered is wrong or taken try again");
                                                        System.out.println("what time do want
to start playing at in  24 hours format ");
                                                        startingTime= scanner1.nextInt();
                                                        System.out.println("hat time do want to
end the match at in  24 hours format");
                                                        endingTime= scanner1.nextInt();
                                                    }
                                                    System.out.println("you has booked the
playground successfully");

currentPlayer.addBookingID(i.getLastBookingID());
                                                    break;

                                                }
                                                break;
                                            }

                                        } else
                                            break;
                                    }catch (InputMismatchException e){
                                        System.out.println("you should have entered a int
value");
                                    }
                                }
                            }else if(int_input==3){

                            }else if(int_input==4){

                            }else if(int_input==5){
                                break;

                            }else {
                                System.out.println("you entered a wrong input");

                            }



                        } catch (InputMismatchException e) {
                            System.out.println("sorry you entered a wrong input  \n");
                            scanner1.nextLine();
                        }

                    }
```

Software Design Specification

```
            }else if(int_input==2){
                while(true){
                    if(!currentPlayer.getBookingsID().isEmpty()){
                        List<Booking> bookings= currentPlayer.getMyBookings();
                        for(Booking i: bookings){
                            i.displayForPlayer();
                            System.out.println("\n");
                        }
                        System.out.println("[0] back");
                        string_input =scanner.nextLine();
                        break;



                    }else {
                        System.out.println("you dont have any bookings");
                    }
                }


            }else if(int_input==3){
                break;

            }else {
                System.out.println("you entered a wrong value");
            }

        }catch (InputMismatchException e){
            System.out.println("please enter an integer value");
            scanner.nextLine();
            scanner1.nextLine();
        }
    }

}

/**
 * static method allow the user to mange every thing about his team and the teams he joined
 * @param currentPlayer the player who recently logged in or signed up
 */
public static void mangeTeams(Player currentPlayer){
    Scanner scanner = new Scanner(System.in);
    Scanner scanner1 = new Scanner(System.in);
    String string_input;
    int int_input = 0;
    while(true) {
        try {

            System.out.println("[1] create team");
            System.out.println("[2] my team");
            System.out.println("[3] my joined teams");
            System.out.println("[4] back");
            int_input = scanner1.nextInt();
            if (int_input == 1){
```

Software Design Specification

```
                    if(currentPlayer.getMyTeamID().equals("none")) {
                        Team currentTeam = new Team();

                        System.out.println("enter team ID it (must be unique one) :  ");
                        string_input = scanner.nextLine();
                        while (!currentTeam.signUpID(string_input)) {
                            System.out.println("the ID you entered is not valid or maybe
taken");

                            string_input = scanner.nextLine();

                        }
                        System.out.println("accepted \n Team ID is : " + string_input);
                        String TeamID = string_input;
                        //setting the  ID player for the team
                        currentTeam.setTeamLeaderID(currentPlayer.getID());
                        // adding the to data base
                        currentPlayer.addMyTeam(currentTeam);
                        System.out.println("Team created successfully");
                        System.out.println("you can go to mange invitations and invite your
friends");
//                          DataSource.getInstance().printTeams();


                    }else{
                        System.out.println("you already created your team");
                    }

                }else if(int_input==2){

                    if(currentPlayer.getMyTeam()==null) {
                        System.out.println("you Dont have any Team");
                        continue;
                    }

                    currentPlayer.getMyTeam().display();
                    System.out.println("\n\n [0] back");
                    string_input=scanner.nextLine();
                    break;


                }else if(int_input==3){

                    List<String> joinedTeamsIDs = currentPlayer.getJoinedTeamsIDs();
                    if(!joinedTeamsIDs.isEmpty()) {
                        System.out.println(" Teams you join  :");


                        for (String i : joinedTeamsIDs) {
                            DataSource.getInstance().getTeamByID(i).display();
                        }

                        System.out.println("[0] back");
                        string_input = scanner.nextLine();


                    }else
```

Software Design Specification

```
                        System.out.println("you did not join any team");
                }else if(int_input==4){

                    break;

                } else{
                    System.out.println("you entered wrong input");
                }
            }catch (InputMismatchException e){
                System.out.println("sorry you entered a wrong input  \n");
                scanner1.nextLine();

            }
        }
    }

    /**
     * static method for the player to interact with the system that allow him the mange his
invitations
     * so he can send an invitation or accept one or maybe reject it
     * @param currentPlayer the player who recently logged in or signed up
     */
    public static void mangeInvitations(Player currentPlayer){
        while(true) {

            Scanner scanner = new Scanner(System.in);
            Scanner scanner1 = new Scanner(System.in);
            String string_input;
            int int_input = 0;
            try {

                System.out.println("[1]send invitation to other player to join your team");
                System.out.println("[2] view sent invitation");
                System.out.println("[3] back");

                int_input = scanner1.nextInt();
                if (int_input == 1) {
                   if(!currentPlayer.getMyTeamID().equals("none")) {
                       Invitation newInvitation = new Invitation();
                       newInvitation.setSenderID(currentPlayer.getID());
                       newInvitation.setTeamInvitedID(currentPlayer.getMyTeam().getID());
                       while (true) {
                           System.out.println("enter the Player's ID you want to invite");
                           string_input = scanner.nextLine();
                           while (!newInvitation.checkReceiverID(string_input)) {
                               System.out.println("the ID you entered is not right try again");
                               string_input = scanner.nextLine();

                           }
                           newInvitation.addReceiver(string_input);
                           System.out.println("sent anther invitation? (y/n)");
                           string_input = scanner.nextLine();
                           if (string_input.equals("y") || string_input.equals("Y"))
                               continue;
                           System.out.println("enter the massage attached by the invitation
:");
```

Software Design Specification

```
                            string_input = scanner.nextLine();
                            newInvitation.setMassage(string_input);
                            currentPlayer.sendInvitation(newInvitation);

                            break;
                        }
                    }else
                        System.out.println("sorry you did not create any team");
                             break;

                } else if (int_input == 2) {
                    ArrayList<Invitation> sentInvitations = currentPlayer.receiveInvitations();
                    if(!sentInvitations.isEmpty()) {
                        for (Invitation i : sentInvitations) {

                            i.display();
                            System.out.println("\n");
                        }while(true) {
                            int check1=0;
                            System.out.println("enter the invitation ID you the team you want
to join");
                            string_input = scanner.nextLine();
                            for (Invitation i : sentInvitations) {
                                if (string_input.equals(i.getID())) {
                                    //adding the team ID to the player
                                    currentPlayer.addJoinedTeam(i.getTeamInvitedID());
                                    //getting the team it self from the data scource and and
add the player ID to the joined players

DataSource.getInstance().getTeamByID(i.getTeamInvitedID()).addInvitedPlayer(currentPlayer.getID
());
                                    System.out.println("you have been added to  " +
i.getTeamInvitedID() +" team");
                                    i.deleteReceiverByItsID(currentPlayer.getID());

                                    check1=1;
                                    break;
                                } else {
                                    System.out.println("you entered a wrong ID try again");
                                }

                            }
                            if(check1==1)
                            break;
                        }


                    }else {
                        System.out.println("you don't have any invitations yet");
                    }
                }else if (int_input==3){
                    break;
                }


            } catch (InputMismatchException e) {
```

Software Design Specification

```
                System.out.println("sorry you entered a wrong input  \n");
                scanner1.nextLine();


            }
        }



    }

    /**
     * allow the user to update his name or email or his password
     * @param currentPlayer the player who recently logged in or signed up
     */
    public static void updateInfo(Player currentPlayer){
        Scanner scanner = new Scanner(System.in);
        Scanner scanner1 = new Scanner(System.in);
        String string_input;
        int int_input = 0;
        while(true) {
            try {
                System.out.println("your current ID is : " + currentPlayer.getID());

                System.out.println("your current name is : " + currentPlayer.getName());
                System.out.println("your Current Email is : " + currentPlayer.getEmail() +
"\n");


                System.out.println("your ID can not be updated .\n");

                System.out.println("[1] change name ");
                System.out.println("[2] change Email");
                System.out.println("[3] change password");
                System.out.println("[4] back");
                int_input = scanner1.nextInt();
                if(int_input==1){
                    System.out.println("enter the new name :");
                    string_input= scanner.nextLine();
                    while (!currentPlayer.signUp_name(string_input)){
                        System.out.println("the name you entered is not valid ");
                        string_input= scanner.nextLine();
                    }

                    System.out.println("name changed successfully");

                }else if(int_input==2){
                    System.out.println("enter you new email");
                    string_input= scanner.nextLine();
                    while (!currentPlayer.signUp_Email(string_input)){
                        System.out.println("the email you entered is not valid ");
                        string_input= scanner.nextLine();

                    }
                    System.out.println("Email changed successfully");

                }else if(int_input==3){
```

Software Design Specification

```
                    System.out.println("enter you last password : ");
                    string_input=scanner.nextLine();
                    while(!string_input.equals(currentPlayer.getPassWord())){
                        System.out.println("the password you entered is wrong try again ");
                        string_input=scanner.nextLine();

                    }
                    System.out.println("enter your new passWord :");
                    string_input = scanner.nextLine();
                    while(!currentPlayer.signUp_password(string_input)){
                        System.out.println("this password is not valid ");
                        string_input =scanner.nextLine();
                    }

                }else if(int_input==4){
                    break;

                }else
                    System.out.println("wrong input");
            }catch (InputMismatchException e){
                System.out.println("enter a integer value");
                scanner1.nextLine();
            }
        }



    }


}
```

I_LogIn interface :

```
package GOFO.Registering;

import GOFO.Users.User;

/**
 * interface contain all the method needed for for Logging in
 * @author abdraham fesal
 */
public interface I_LogIn {
    boolean checkInfo(String ID,String password);
   User logIn();

}
```

I_SignUp interface :

Software Design Specification

```
package GOFO.Registering;

/**
 * interface contain all the method needed for for signing up
 *  * @author mohamed mahmoud
 */
public interface I_SignUp {

    boolean signUp_name(String name);

    boolean signUp_ID(String ID);

    boolean signUp_Email(String Email);
    boolean signUp_password(String password);
    void create_account ();

}
```

I_UdataInfo interface :

```
package GOFO.Registering;

/**
 * interface contain all the method needed for Updating info
 * @author abdraham fesal
 */
public interface I_UdataInfo {
    boolean chanceName(String name);
    boolean chancePassword(String password);
}
```

Main class :

```
package GOFO;

import GOFO.contorllers.AdminProfile;
import GOFO.contorllers.HomeScreen;
import GOFO.contorllers.OwnerProfile;
import GOFO.DataModel.DataSource;
import GOFO.contorllers.PlayersProfile;
import GOFO.Users.Admin;
import GOFO.Users.Owner;
import GOFO.Users.Player;
import GOFO.Users.User;

import java.util.InputMismatchException;
import java.util.Scanner;

/**
 * the Main class the point the program starts at and it is a console based porgram
 * @author  mohamed mahmoud
```

Software Design Specification

```
 */
public class Main {

    public static void main(String[] args) {
        DataSource.getInstance().hardCodedUsers();
        Scanner scanner = new Scanner(System.in);
        Scanner scanner1 = new Scanner(System.in);
        int check=0;


        String string_input;
        int int_input = 0;
        Player currentPlayer = null;
        Owner currentOwner = null;
        User currentUser =null;
        Admin admin = null;
        while(true) {
            while (true) {
                try {
                   HomeScreen.printMenu();
                    int_input = scanner1.nextInt();
                    if (int_input==1){
                        currentUser = HomeScreen.userLogIn();
                        break;
                    } else if (int_input == 2) {
                        currentUser= HomeScreen.playerSigningUp();
                        break;
                    } else if (int_input == 3) {
                        currentUser = HomeScreen.ownerSigningUp();
                        break;
                    } else {
                        System.out.println("please enter a valid integer input ");
                    }
                } catch (InputMismatchException e) {
                    System.out.println("sorry you entered a wrong input  \n");
                    scanner1.nextLine();
                }
            }
            while(true){
                if(currentUser.getType().equals("Player")){
                    while(true) {
                        try {

                            PlayersProfile.printMenu((Player) currentUser);
                            int_input = scanner1.nextInt();
                            if (int_input == 1) {
                                PlayersProfile.mangeBooking((Player) currentUser);

                            } else if (int_input == 2) {
                                PlayersProfile.mangeTeams((Player) currentUser);
                            } else if (int_input == 3) {
                                PlayersProfile.mangeInvitations((Player) currentUser);
                            } else if (int_input == 4) {
                                PlayersProfile.updateInfo((Player) currentUser);

                            } else if (int_input == 5) {
```

Software Design Specification

```
                        check=1;
                        break;


                    } else {
                        System.out.println("wrong input !");

                    }

                } catch (InputMismatchException e) {
                    System.out.println("sorry you entered a wrong input  \n");
                    scanner1.nextLine();


                }
            }
            if(check==1){
                check=0;
                break;

            }


        }else if(currentUser.getType().equals("Owner")){
            while(true) {
                try {



                OwnerProfile.printMenu((Owner) currentUser);
                int_input = scanner1.nextInt();
                if (int_input == 1) {
                    OwnerProfile.viewPlayGrounds((Owner) currentUser);

                } else if (int_input == 2) {
                    OwnerProfile.AddNewPlayGround((Owner) currentUser);

                } else if (int_input == 3) {

                } else if (int_input == 4) {

                } else if (int_input == 5) {
                    check=1;
                    break;

                } else {
                    System.out.println("wrong input !");


                }

            }catch (InputMismatchException e){
                    System.out.println("wrong input !");
                    scanner1.nextLine();

                }
```

Software Design Specification

```
                    }
                    if(check==1){
                        check=0;
                        break;
                    }


                }else if(currentUser.getType().equals("Admin")){
                    while(true) {
                        try {
                            AdminProfile.displayMenu();
                            int_input = scanner1.nextInt();
                            if (int_input == 1) {
                                AdminProfile.suspendPlayGrounds();


                            } else if (int_input == 2) {
                                AdminProfile.activePlayGround();


                            } else if (int_input == 3) {
                                AdminProfile.deletePlayGround();


                            } else if (int_input == 4) {
                                check=1;
                                break;


                            }
                            else {
                                System.out.println("wrong input !");


                            }
                            }catch (InputMismatchException e){
                            System.out.println("wrong input");
                            scanner1.nextLine();
                        }

                    }
                    if(check==1){
                        check=0;
                        break;
                    }

                }

            }

        }
    }
}
```

CS251: Phase 2 – <Team Name>
Project: <GOFO>

Software Design Specification