Cairo University
Faculty of Computers and Artificial Intelligence

CS251

Software Engineering I

GoFo

Final Project

Project ID: PM-988

June & 2020

Contents

Team

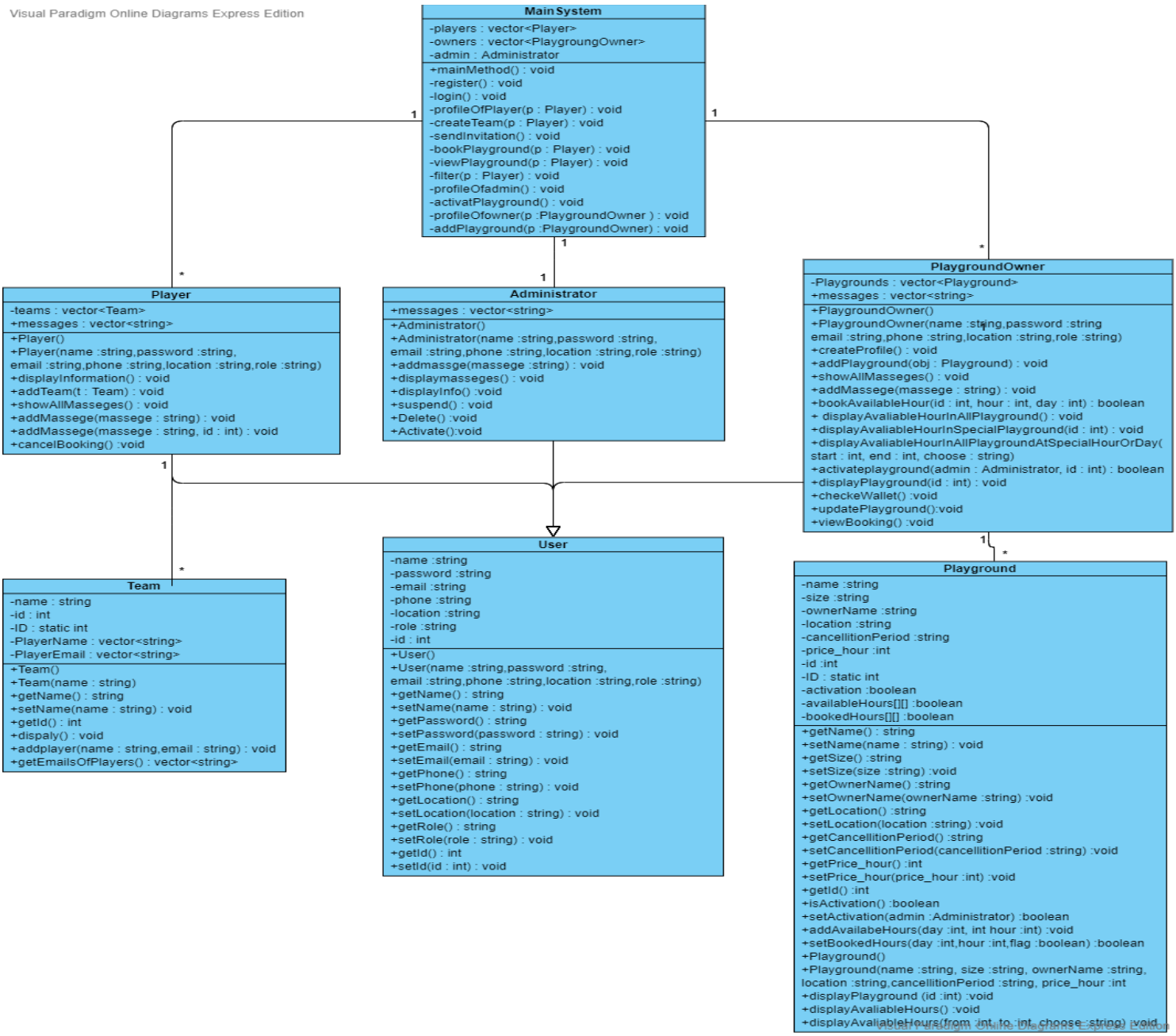
ID	Name	Email	Mobile
20180205	Kerols Samir Saif	Kerolssamer348@gmail.com	01554860303
20180094	Khaled Osama Elshazly	5aledosamaa123@gmail.com	01011779158
20180242	Mohamed Hany Khairy	Muhmedhaniiii40@gmail.com	01200076622
20180223	Mohamed Saeed Said	saeedmuhammad728@gmail.com	01143354928
20180055	Omnia Fares Elsayed	omniafares133@gmail.com	01129936968

Document Purpose and Audience

- This document includes SDS description for GoFo football playground booking system. It describes the class diagram, sequence diagram, UI and other phases which are needed to complete the project.
- It is meant for all stakeholders, especially the client to understand what features will in the system.
- It also serves as the basis for the contract between the company and the client.
- It serves as a guide for the developers to understand what they will develop.

System Models

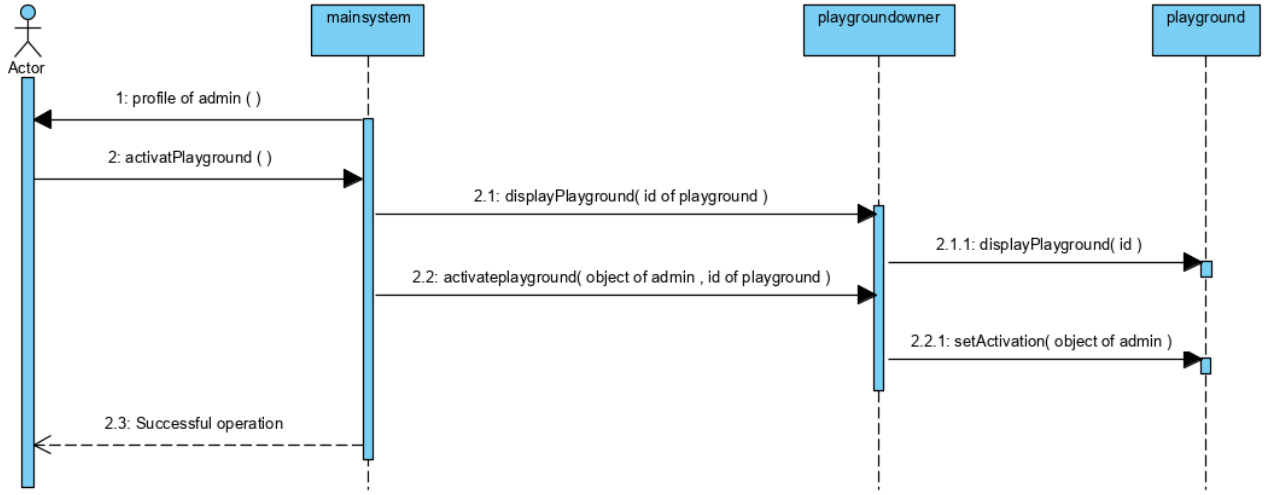
I. Class Diagram(s)

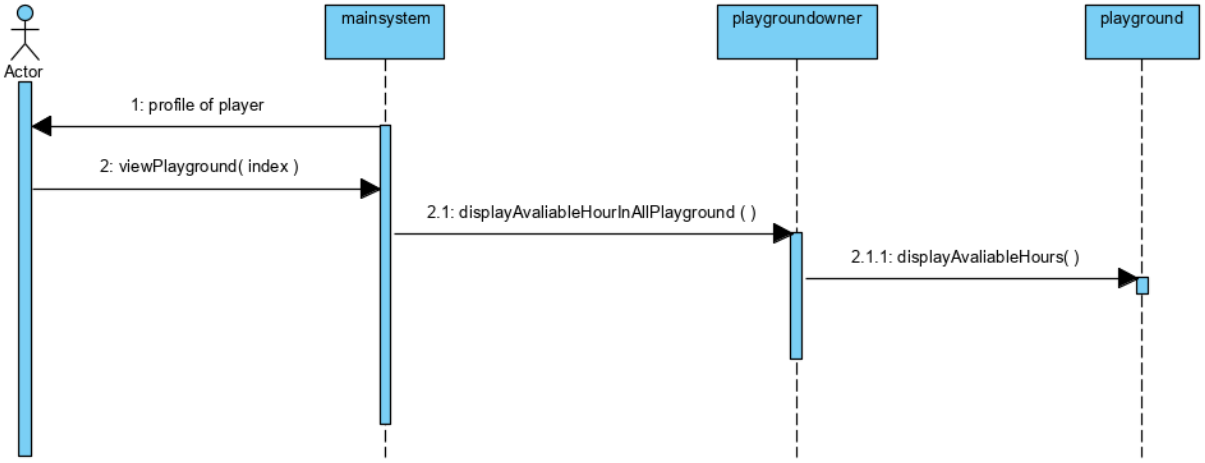


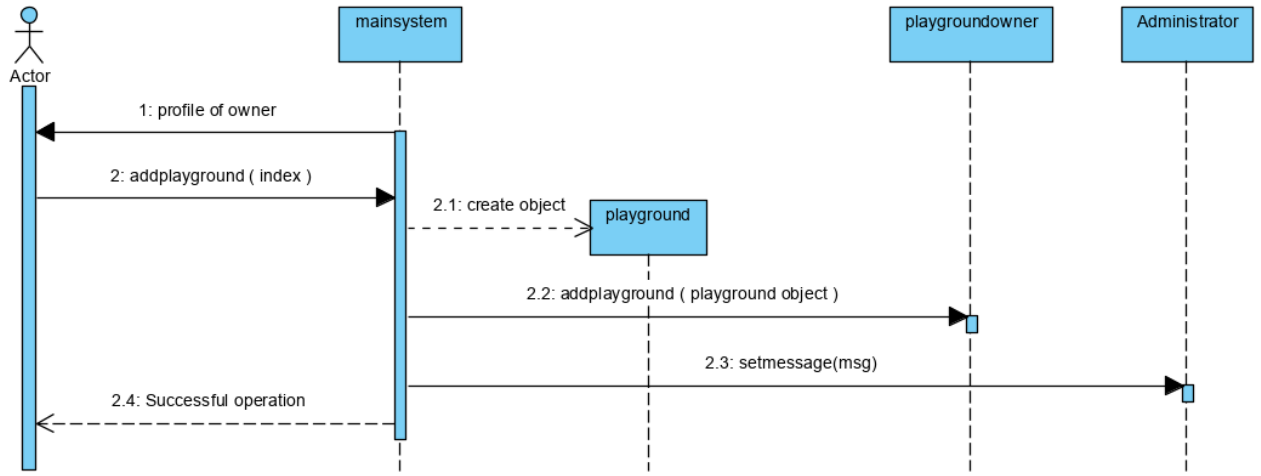
II. Class Descriptions

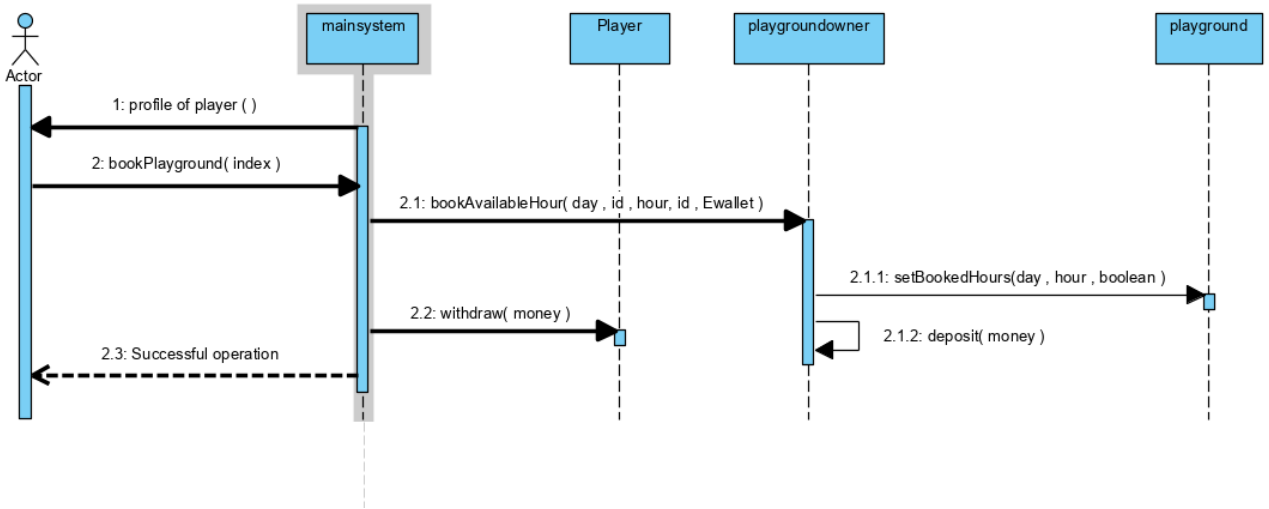
Class ID	Class Name	Description & Responsibility
1.	User	This class related to any user will use this system which includes: the administrator, owner of playground or player. all methods in this class serve this functionality for all of them. attributes in this class (name, email, password, phone, location, id, role) describe the information of the user. And it has setter and getter for each attribute.
2.	Player	This class inhered from user class so it has all methods and attributes in user class plus the features that the player has. Each player has vector contains the teams he joined in and a vector of messages he receives. The responsibility of each object from this class is adding team, adding message, display all the player information, show all messages and cancel a certain booking if he wants.
3.	Team	As we described that each player can have one or more than one team, this class has the information about the team such as the (name of team, team id, names and emails of all the players joining the team) and has the responsibility of setting and getting these attributes as well as adding players in the teams and showing the details of the team.
4.	PlaygroundOwner	This class is inhered from user class so it has the ability to use all methods in it plus the features that the playground owner has. each object has vector contains all the playgrounds the owner has and all the messages he receives. Each playground owner from this class has the ability to show his entire information, add or show messages, add or update or show playground, show the available hours for booking in all the playgrounds or in certain playground, ask the administrator to activate his playground, check eWallet for payment and view booked playgrounds with its booked hours.
5.	Playground	This class holds the playgrounds owned by the playground owner as any owner can have any number of playgrounds. This class is responsible for setting and getting all the data related to the playground such as its name, id, size, location, checking if this playground is active or not, adding available hours and display them and display the entire information about the playground.
6.	Administrator	This class for the administrator who oversees the overall operations of the system and ensures that no fraud takes place and this system has one administrator and this class is inhered from user class so it has the ability to use all methods in it plus the features that the administrator has. Each object has vector of messages the administrator receives and has the ability to add or show messages, display the admin information and has the right to delete a playground or suspend it or activate it.
7.	MainSystem	This class is responsible for managing the entire system and uniting the other classes together. It has any number of players and owners but has only one admin. It is responsible for the main functions such as register, login, player menu, admin menu, owner menu, filtering or adding or activating or viewing or booking playground, creating teams and send them invitations.

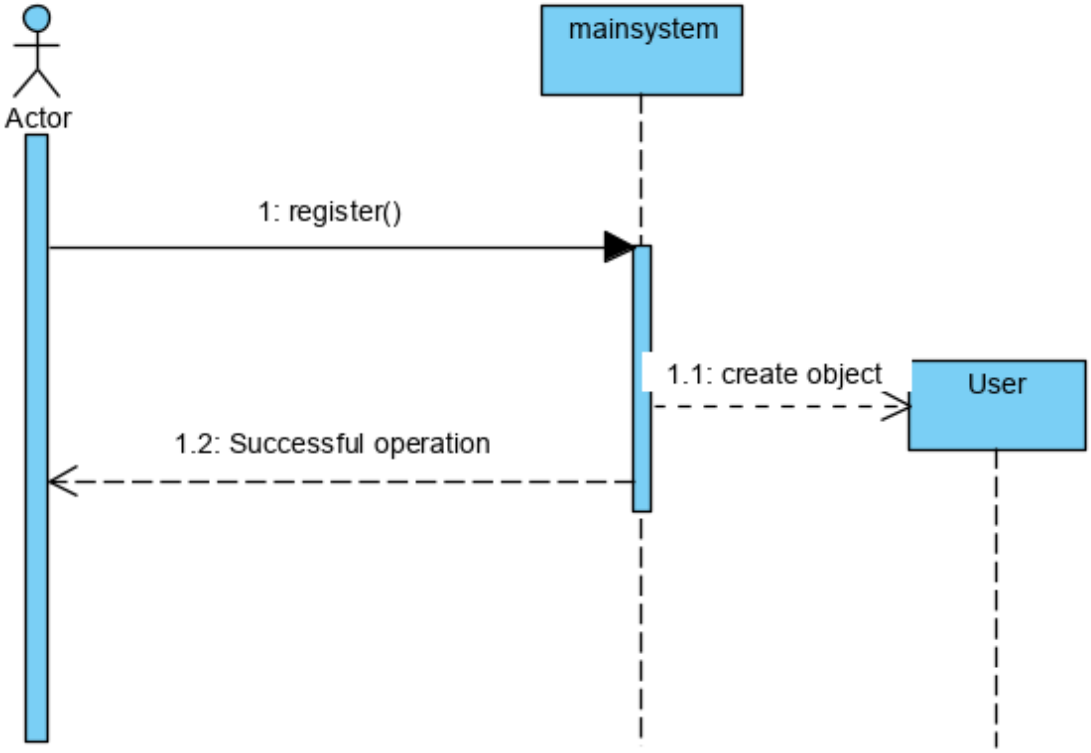
III. Sequence diagrams

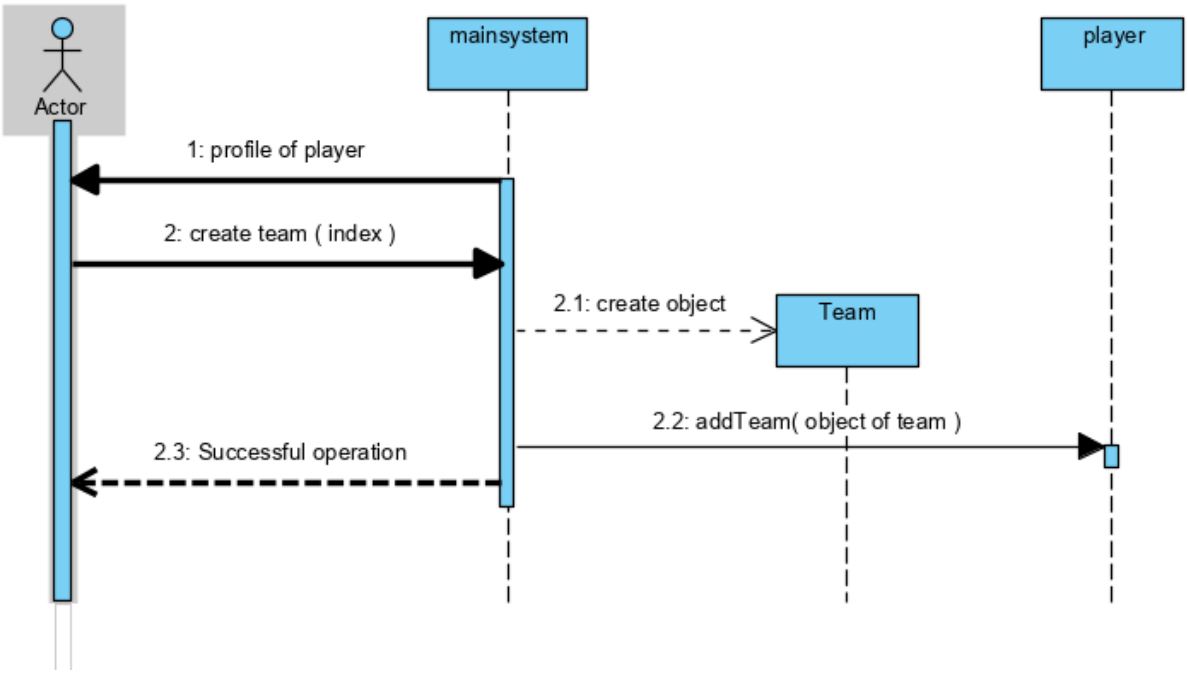





Actor
mainsystem
playgroundowner
Administrator
1: profile of owner
2: addplayground (index)
2.1: create object
playground
2.2: addplayground (playground object)
2.3: setmessage(msg)
2.4: Successful operation


Actor
mainsystem
Player
playgroundowner
playground
1: profile of player ( )
2: bookPlayground(index)
2.1: bookAvailableHour(day , id , hour, id , Ewallet)
2.1.1: setBookedHours(day , hour , boolean)
2.2: withdraw(money)
2.1.2: deposit(money)
2.3: Successful operation


Actor
mainsystem
1: register()
1.1: create object
User
1.2: Successful operation


Actor
mainsystem
player
1: profile of player
2: create team (index)
2.1: create object
Team
2.2: addTeam(object of team)
2.3: Successful operation

Class - Sequence Usage Table

Sequence Diagram	Classes Used	All Methods Used
1. Register user	Main system User	Register()

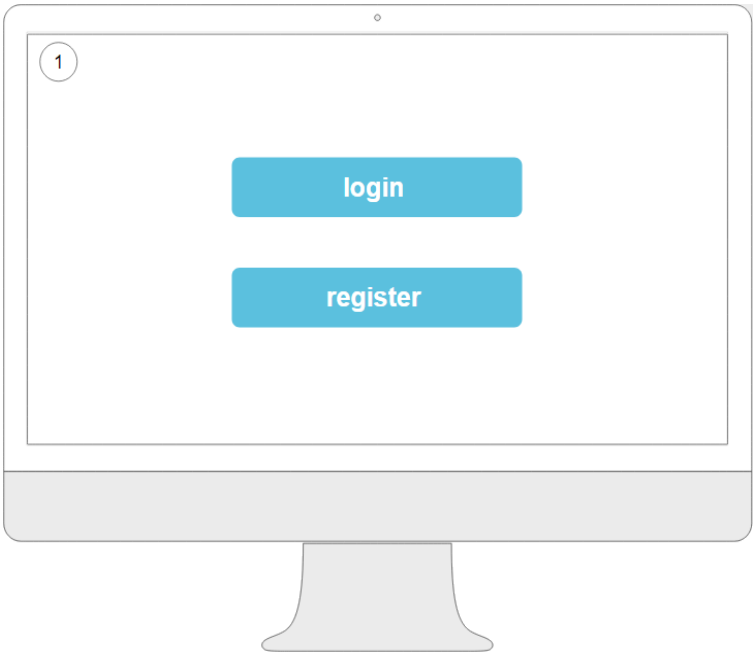
2. Add playground	Main system
	Playground owner
	Playground
	Administrator
		Profile of owner()
		Add playground() //owner
		Add playground() //playground
		Set massage()

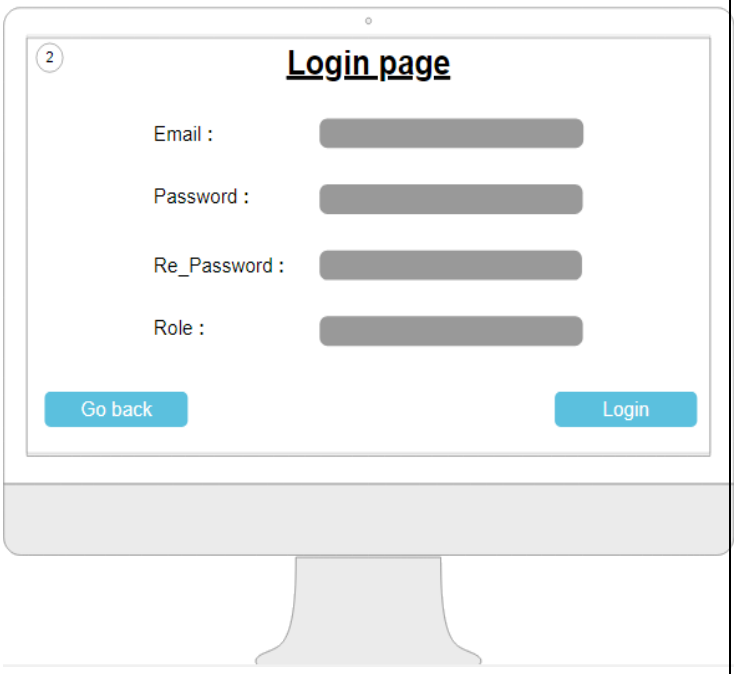
3. Book playground	Main system Playground owner Playground Player	Profile of player() Book playground() Book available hour() Set booked hours() Deposit() Withdraw()
4. View playing hours	Main system Playground owner Playground	Profile of player() View playground() Display available hours in all playground() Display available()

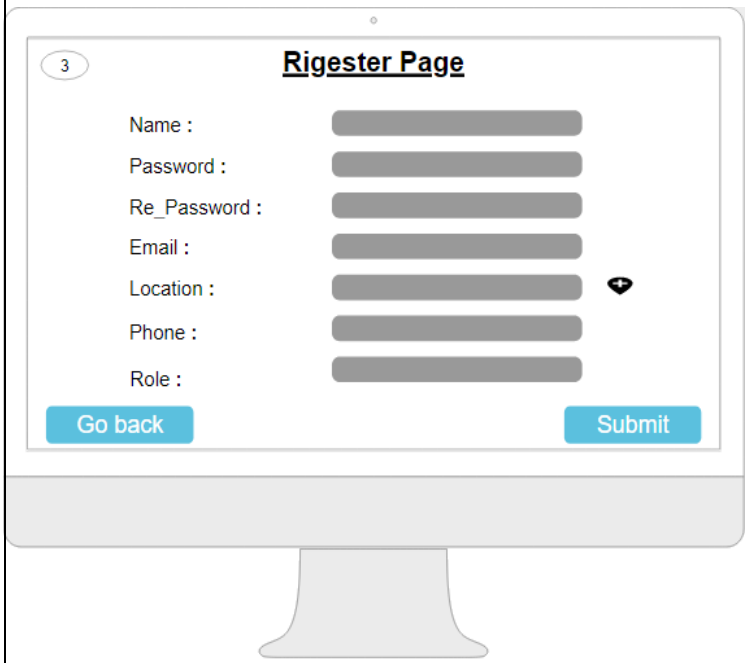
5. Approve playground	Main system
	Playground owner
	Playground
		Profile of admin()
		Activate playground() //main system
		Display playground() //owner
		Display playground() //playground
		Activate playground() //owner
		Set activation()

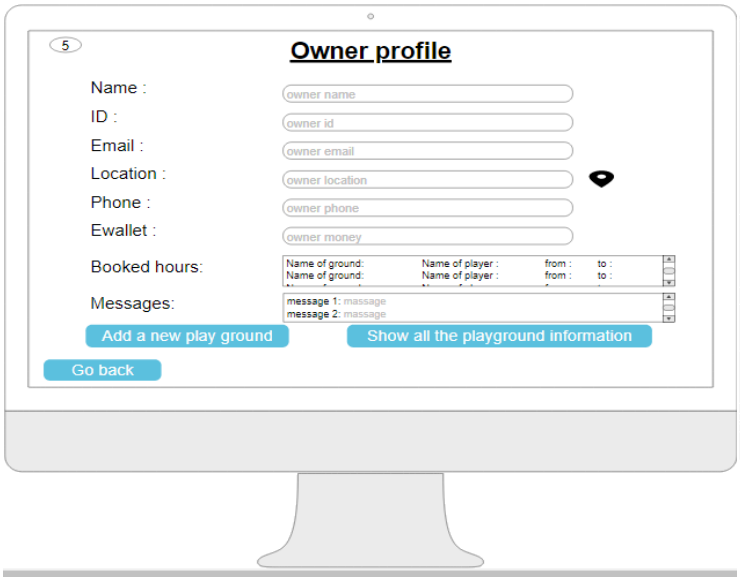
6. Create a team	Main system Player Team	Profile of player() Create team() Add team()

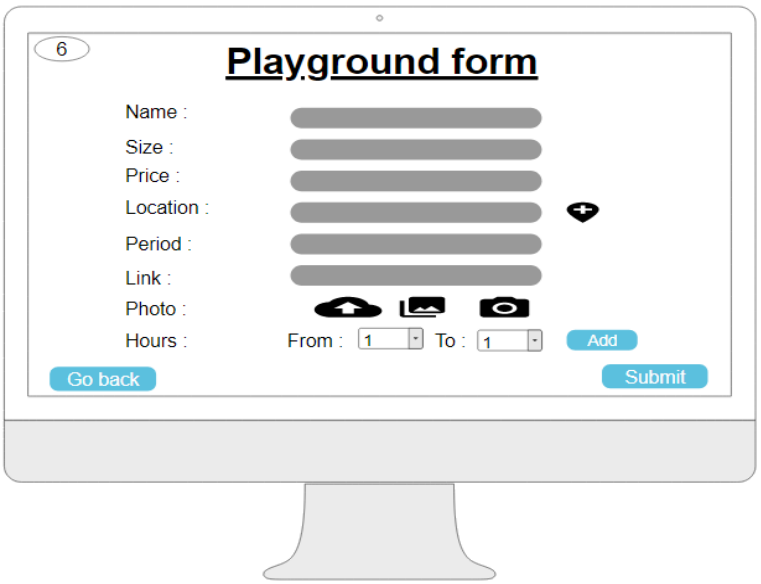
IV. User Interface Design

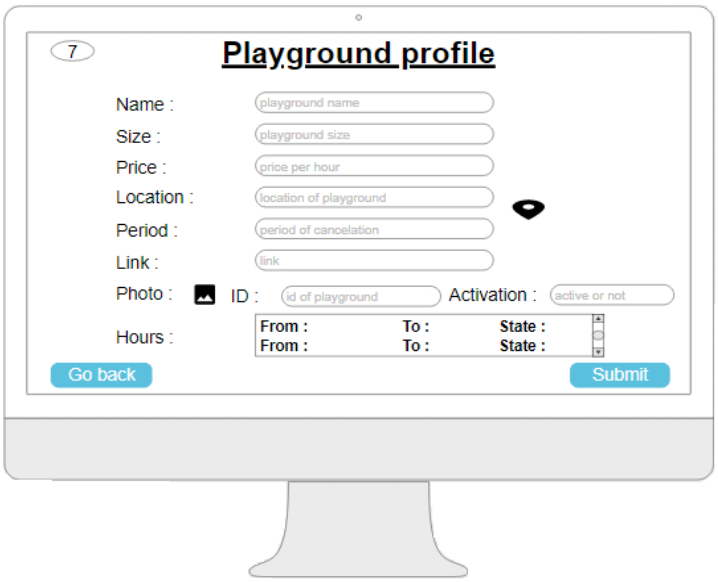




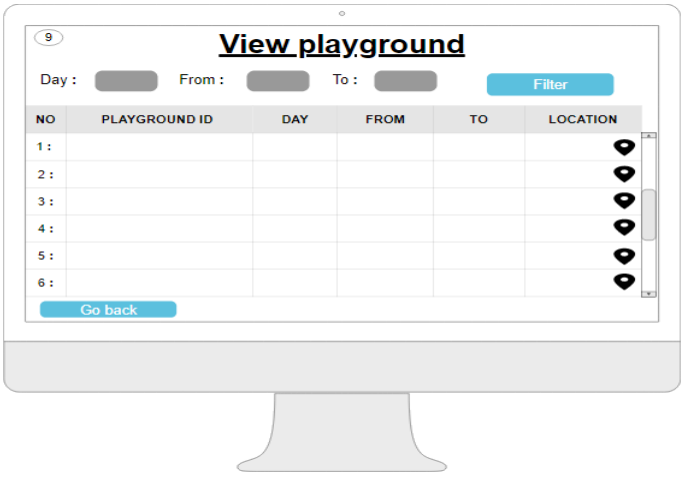


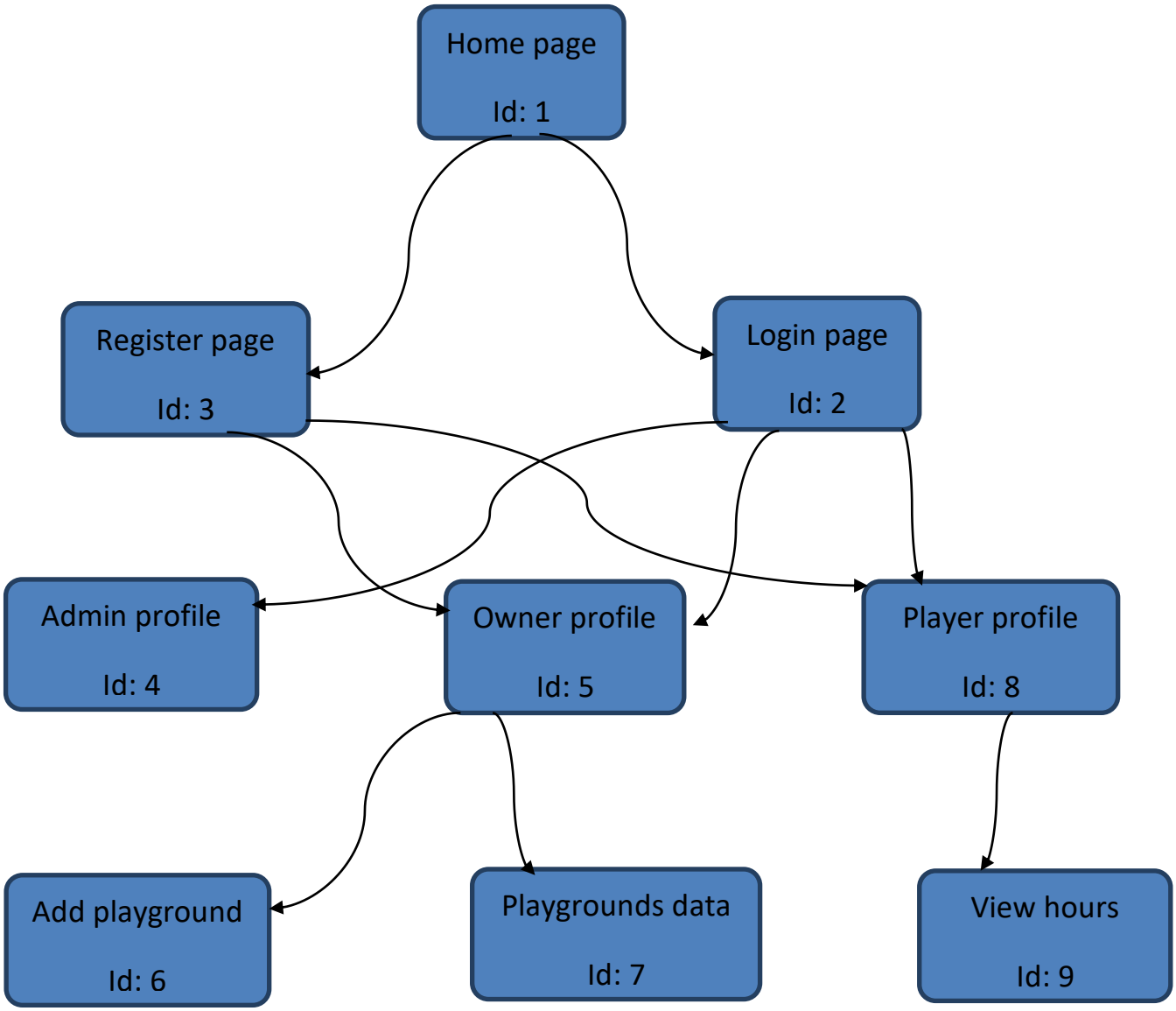

5
Owner profile
Name :
ID :
Email :
Location :
Phone :
Ewallet :
Booked hours:
Messages:
owner name
owner id
owner email
owner location
owner phone
owner money
Add a new play ground
Show all the playground information
Go back


6
Playground form
Name :
Size :
Price :
Location :
Period :
Link :
Photo :
Hours :
From :
To :
Add
Go back
Submit


7
Playground profile
Name :
Size :
Price :
Location :
Period :
Link :
Photo :
ID :
Activation :
Hours :
playground name
playground size
price per hour
location of playground
period of cancelation
link
id of playground
active or not
From : To : State :
From : To : State :
Go back
Submit


8
Player profile
Name :
ID :
Email :
Location :
Phone :
Messages:
owner name
owner id
owner email
owner location
owner phone
Create team
Name of team :
Name of player :
Add
Massage :
Team
player
ID :
Send
ID of playground :
From :
To :
Make booking
Cancel booking
View playground
Go back


9
View playground
Day :
From :
To :
Filter
NO
PLAYGROUND ID
DAY
FROM
TO
LOCATION
1 :
2 :
3 :
4 :
5 :
6 :
Go back


Home page
Id: 1
Register page
Id: 3
Login page
Id: 2
Admin profile
Id: 4
Owner profile
Id: 5
Player profile
Id: 8
Add playground
Id: 6
Playgrounds data
Id: 7
View hours
Id: 9

Screen ID	Screen Name	Screen / Wireframe Description
1	Home page	User choose to login or register
2	Login	User enter email ,password and role
3	Register	User enter his data correctly
4	Admin profile	This page display all data of admin and he became able to do all the functions (active , delete , approve , suspend)
5	Owner profile	This page display all data of owner and make him able to do some functions (see all booked hours , see the E wallet)
6	Add playground	Make owner to add a playground by input all data of it
7	Playground data	Display the data of playground
8	Player profile	This page display all data of player and make him able to do some functions (create team , send messages (complain , invitation) , make booking , cancel booking)
9	View hours	Make player to view all available hour to make booking with correct information

Tools

- **Eclipse**
- **Visual Paradigm**
- **online Moqups**
- **javaDoc**

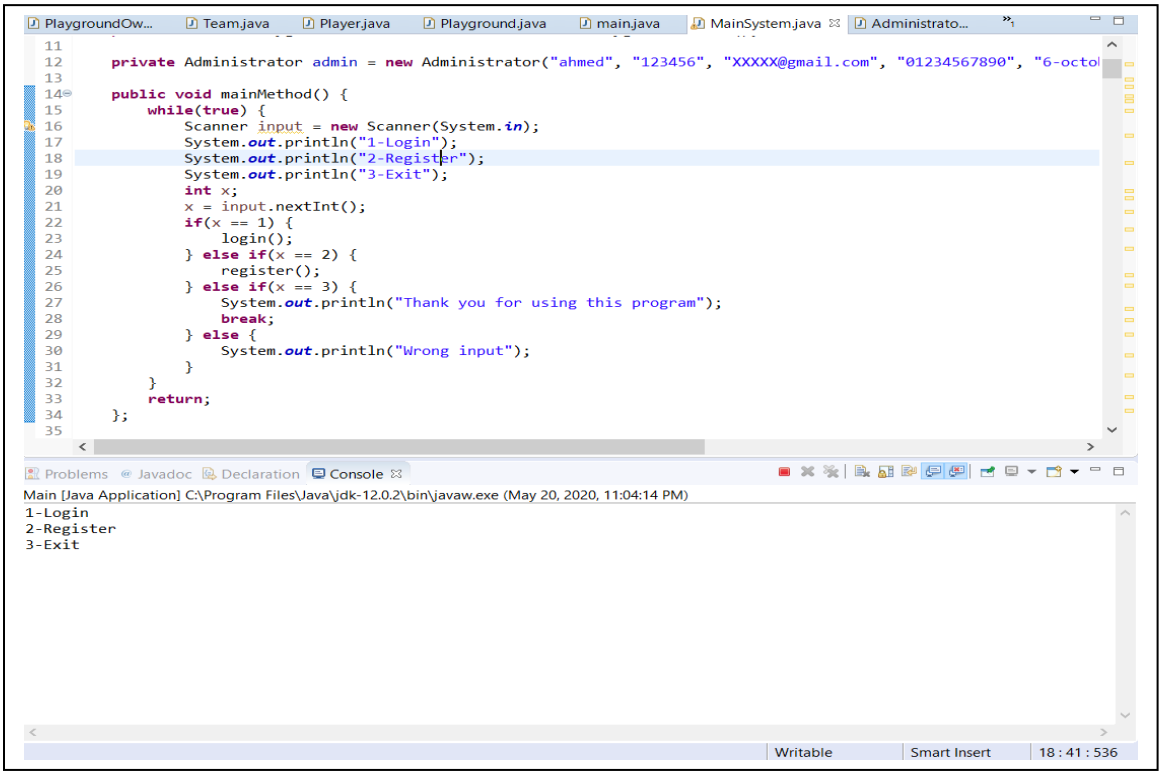
Ownership Report

Item	Owners
Class Diagram	Omnia Fares Khaled Osama
Sequence Diagram	*Kerlos Samir* *Mohamed Saeed*
User Interface Design	*Mohamed Hany* *Kerols Samir*
Implementation	*-Administrator class: Mohamed Hany* *-Player class: Kerlos Samir* *-Playground class: Mohamed Saeed* *-Playground owner class: Omnia Fares* *-Team class: Khaled Osama* *-User class: Kerols Samir* *-Main system class: All of team members (each team member has done 3 functions)*
Github repo	*Mohamed Hany*
Google drive link	*Omnia Fares*
Screen Shots	*Khaled Osama*
Code Listing	*Mohamed Saeed*
Java Doc	*Kerols Samir*

References

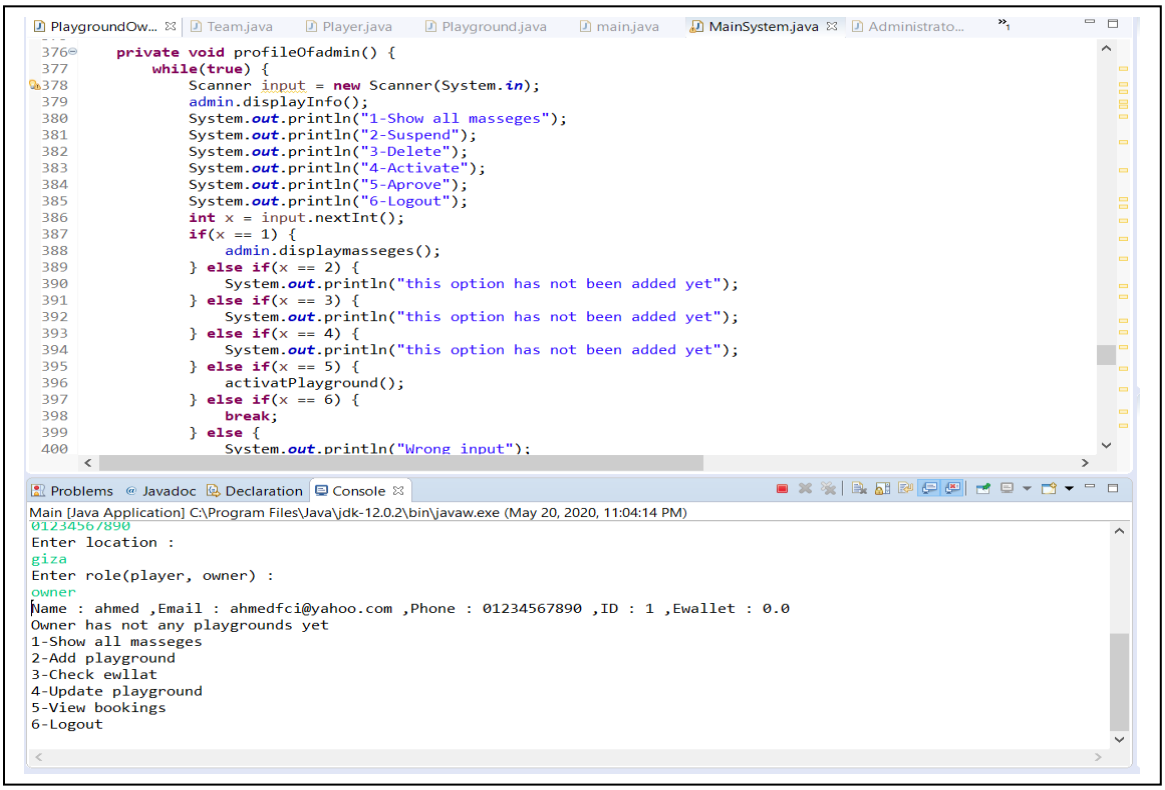
- http://www.mhhe.com/engcs/compsci/pressman/graphics/Pressman5sepa/common/cs1/design.pd
- Mockups (https://moqups.com/).
- How to use Moqups https://www.youtube.com/watch?v=glijkZFo4AY
- Visual paradigm online https://online.visual-paradigm.com/

Appendix A: Code Listing and Screen Snapshots



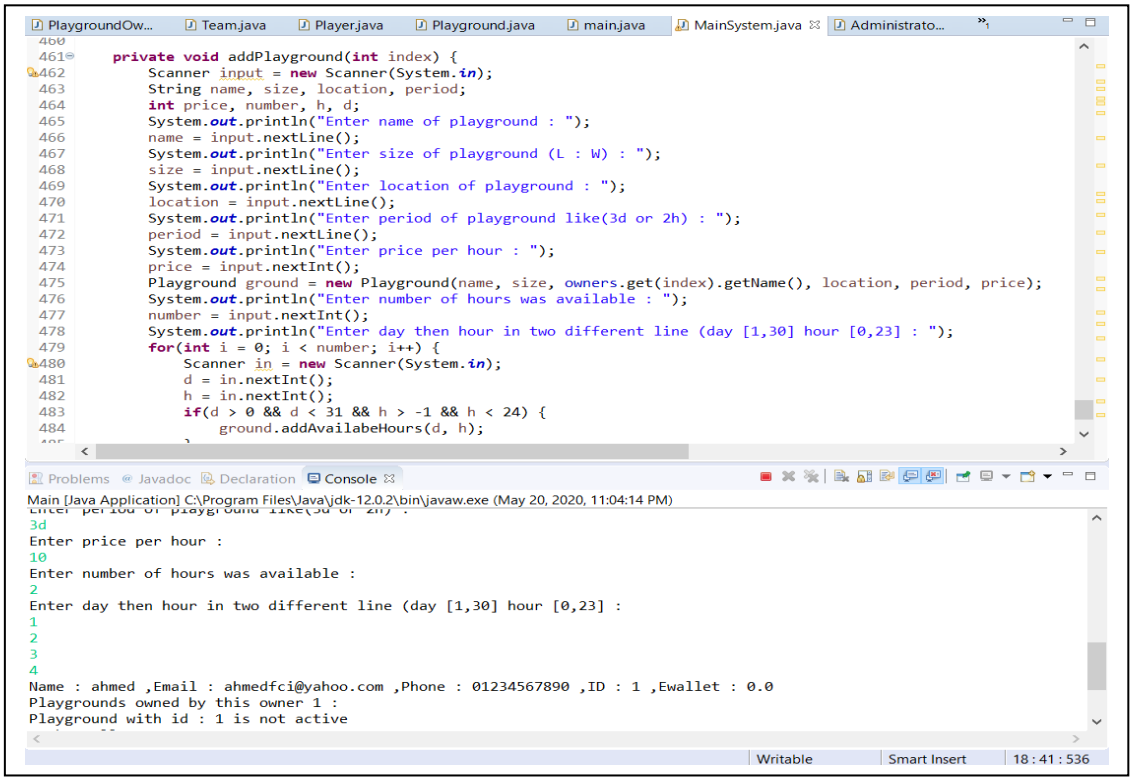
```
11
12      private Administrator admin = new Administrator("ahmed", "123456", "XXXXX@gmail.com", "01234567890", "6-octo
13
14      public void mainMethod() {
15          while(true) {
16              Scanner input = new Scanner(System.in);
17              System.out.println("1-Login");
18              System.out.println("2-Register");
19              System.out.println("3-Exit");
20              int x;
21              x = input.nextInt();
22              if(x == 1) {
23                  login();
24              } else if(x == 2) {
25                  register();
26              } else if(x == 3) {
27                  System.out.println("Thank you for using this program");
28                  break;
29              } else {
30                  System.out.println("Wrong input");
31              }
32          }
33          return;
34      };
35
```

```
Main [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (May 20, 2020, 11:04:14 PM)
1-Login
2-Register
3-Exit
```

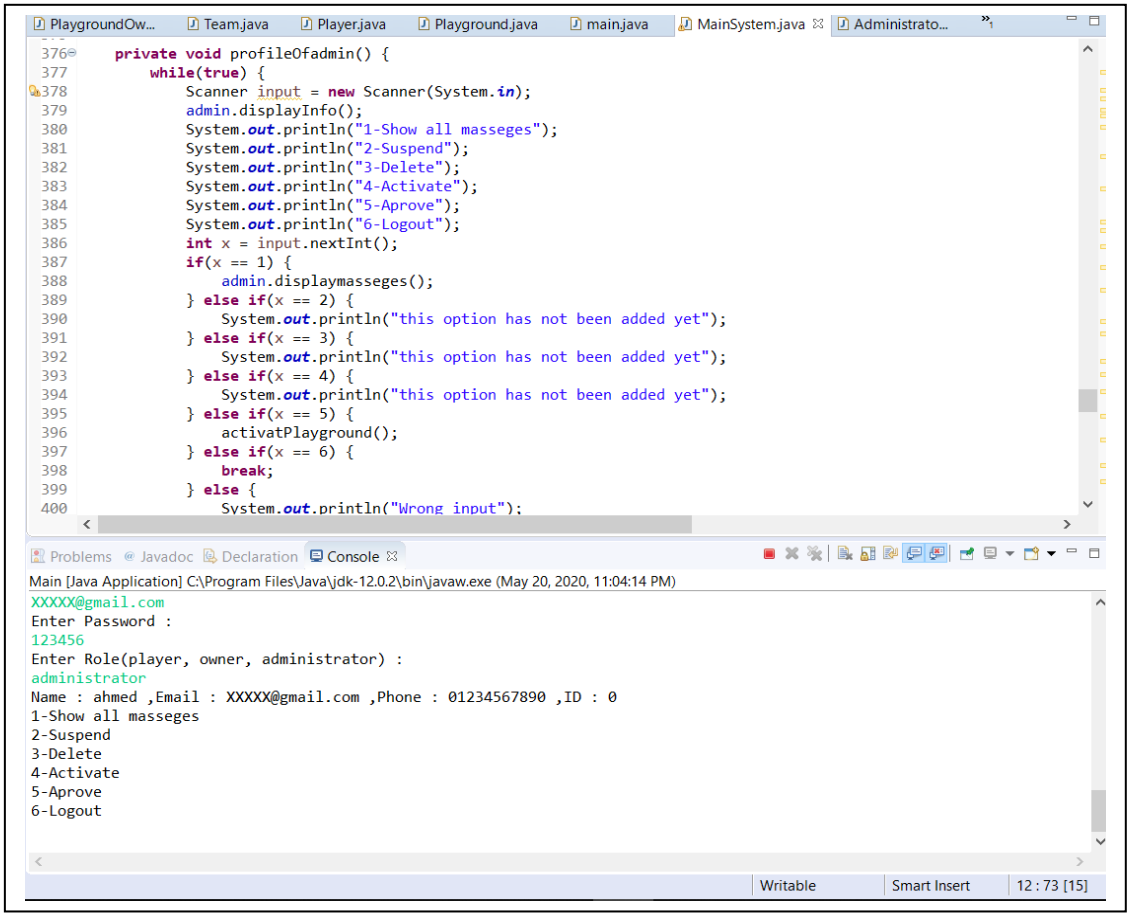


```
376     private void profileOfadmin() {
377         while(true) {
378             Scanner input = new Scanner(System.in);
379             admin.displayInfo();
380             System.out.println("1-Show all masseges");
381             System.out.println("2-Suspend");
382             System.out.println("3-Delete");
383             System.out.println("4-Activate");
384             System.out.println("5-Aprove");
385             System.out.println("6-Logout");
386             int x = input.nextInt();
387             if(x == 1) {
388                 admin.displaymasseges();
389             } else if(x == 2) {
390                 System.out.println("this option has not been added yet");
391             } else if(x == 3) {
392                 System.out.println("this option has not been added yet");
393             } else if(x == 4) {
394                 System.out.println("this option has not been added yet");
395             } else if(x == 5) {
396                 activatPlayground();
397             } else if(x == 6) {
398                 break;
399             } else {
400                 System.out.println("Wrong input");
```

```
Main [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (May 20, 2020, 11:04:14 PM)
01234567890
Enter location :
giza
Enter role(player, owner) :
owner
Name : ahmed ,Email : ahmedfci@yahoo.com ,Phone : 01234567890 ,ID : 1 ,Ewallet : 0.0
Owner has not any playgrounds yet
1-Show all masseges
2-Add playground
3-Check ewllat
4-Update playground
5-View bookings
6-Logout
```



```
PlaygroundOw... Team.java Player.java Playground.java main.java MainSystem.java Administrato...

    private void addPlayground(int index) {
        Scanner input = new Scanner(System.in);
        String name, size, location, period;
        int price, number, h, d;
        System.out.println("Enter name of playground : ");
        name = input.nextLine();
        System.out.println("Enter size of playground (L : W) : ");
        size = input.nextLine();
        System.out.println("Enter location of playground : ");
        location = input.nextLine();
        System.out.println("Enter period of playground like(3d or 2h) : ");
        period = input.nextLine();
        System.out.println("Enter price per hour : ");
        price = input.nextInt();
        Playground ground = new Playground(name, size, owners.get(index).getName(), location, period, price);
        System.out.println("Enter number of hours was available : ");
        number = input.nextInt();
        System.out.println("Enter day then hour in two different line (day [1,30] hour [0,23] : ");
        for(int i = 0; i < number; i++) {
            Scanner in = new Scanner(System.in);
            d = in.nextInt();
            h = in.nextInt();
            if(d > 0 && d < 31 && h > -1 && h < 24) {
                ground.addAvailabeHours(d, h);

Problems Javadoc Declaration Console
Main [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (May 20, 2020, 11:04:14 PM)
3d
Enter price per hour :
10
Enter number of hours was available :
2
Enter day then hour in two different line (day [1,30] hour [0,23] :
1
2
3
4
Name : ahmed ,Email : ahmedfci@yahoo.com ,Phone : 01234567890 ,ID : 1 ,Ewallet : 0.0
Playgrounds owned by this owner 1 :
Playground with id : 1 is not active

Writable Smart Insert 18 : 41 : 536
```



```
PlaygroundOw... Team.java Player.java Playground.java main.java MainSystem.java Administrato...

    private void profileOfadmin() {
        while(true) {
            Scanner input = new Scanner(System.in);
            admin.displayInfo();
            System.out.println("1-Show all masseges");
            System.out.println("2-Suspend");
            System.out.println("3-Delete");
            System.out.println("4-Activate");
            System.out.println("5-Aprove");
            System.out.println("6-Logout");
            int x = input.nextInt();
            if(x == 1) {
                admin.displaymasseges();
            } else if(x == 2) {
                System.out.println("this option has not been added yet");
            } else if(x == 3) {
                System.out.println("this option has not been added yet");
            } else if(x == 4) {
                System.out.println("this option has not been added yet");
            } else if(x == 5) {
                activatPlayground();
            } else if(x == 6) {
                break;
            } else {
                System.out.println("Wrong input");

Problems Javadoc Declaration Console
Main [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (May 20, 2020, 11:04:14 PM)
XXXXX@gmail.com
Enter Password :
123456
Enter Role(player, owner, administrator) :
administrator
Name : ahmed ,Email : XXXXX@gmail.com ,Phone : 01234567890 ,ID : 0
1-Show all masseges
2-Suspend
3-Delete
4-Activate
5-Aprove
6-Logout

Writable Smart Insert 12 : 73 [15]
```

```
405     private void activatPlayground() {
406         Scanner input = new Scanner(System.in);
407         int id1, id2;
408         boolean flag;
409         System.out.println("Enter id of playground : ");
410         id1 = input.nextInt();
411         System.out.println("Enter id of owner : ");
412         id2 = input.nextInt();
413         for(int i = 0; i < owners.size(); i++) {
414             if(owners.get(i).getId() == id2) {
415                 System.out.println("Data of playground : ");
416                 owners.get(i).displayPlayground(id1);
417                 break;
418             }
419         }
420         for(int i = 0; i < owners.size(); i++) {
421             if(owners.get(i).getId() == id2) {
422                 flag = owners.get(i).activateplayground(admin, id1);
423                 if(flag) {
424                     return;
```

Problems Javadoc Declaration Console

Main [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (May 20, 2020, 11:04:14 PM)

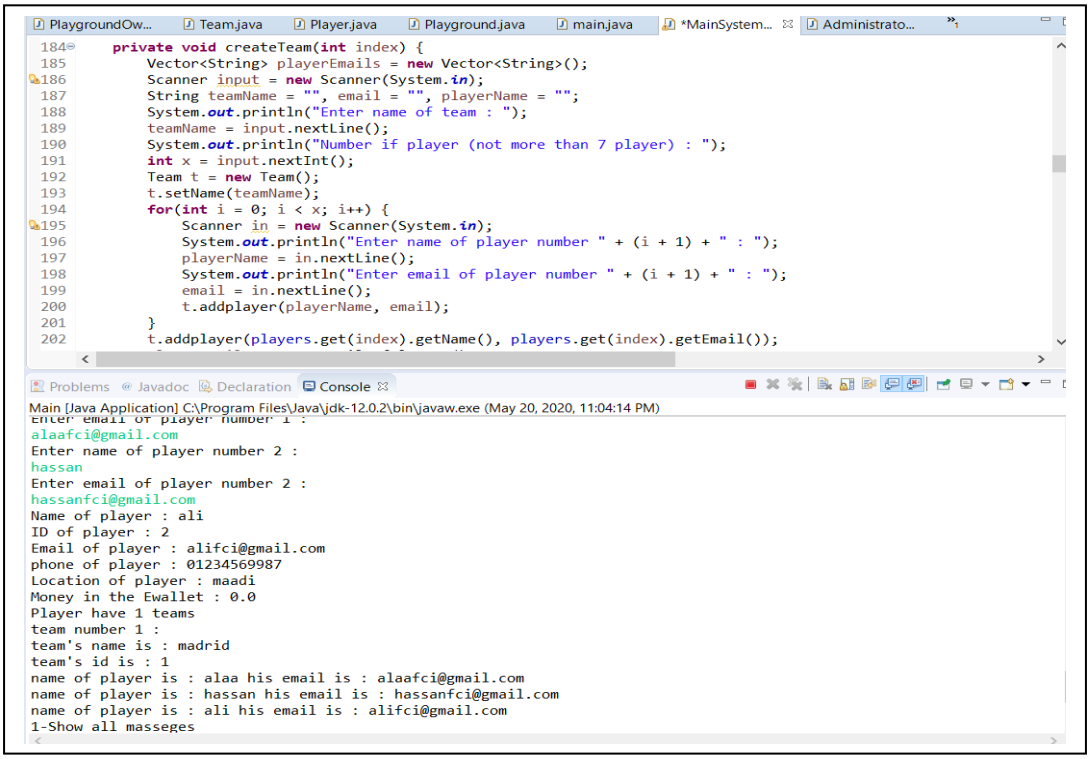
```
1
massege 1 : Owner wit id : 1 make requst to add playground with id : 1
Name : ahmed ,Email : XXXXX@gmail.com ,Phone : 01234567890 ,ID : 0
1-Show all masseges
2-Suspend
3-Delete
4-Activate
5-Aprove
6-Logout
5
Enter id of playground :
1
Enter id of owner :
1
Data of playground :
ID of the playground : 1
Name of the playground : playG1
Size of playground : 100 120
```

```
155             System.out.println("6-View playground");
156             System.out.println("7-eWlallet");
157             System.out.println("8-Logout");
158             int x = input.nextInt();
159             if(x == 1) {
160                 players.get(index).showAllMasseges();
161             } else if(x == 2) {
162                 createTeam(index);
163             } else if(x == 3) {
164                 sendInvitation();
165             } else if(x == 4) {
166                 System.out.println("this option has not been added yet");
167             } else if(x == 5) {
168                 bookPlayground(index);
169             } else if(x == 6) {
170                 viewPlayground(index);
171             } else if(x == 7){
172                 System.out.println("Enter the amount of money : ");
173                 double money;
174                 money = input.nextDouble();
175                 players.get(index).deposit(money);
176             } else if(x == 8){
```

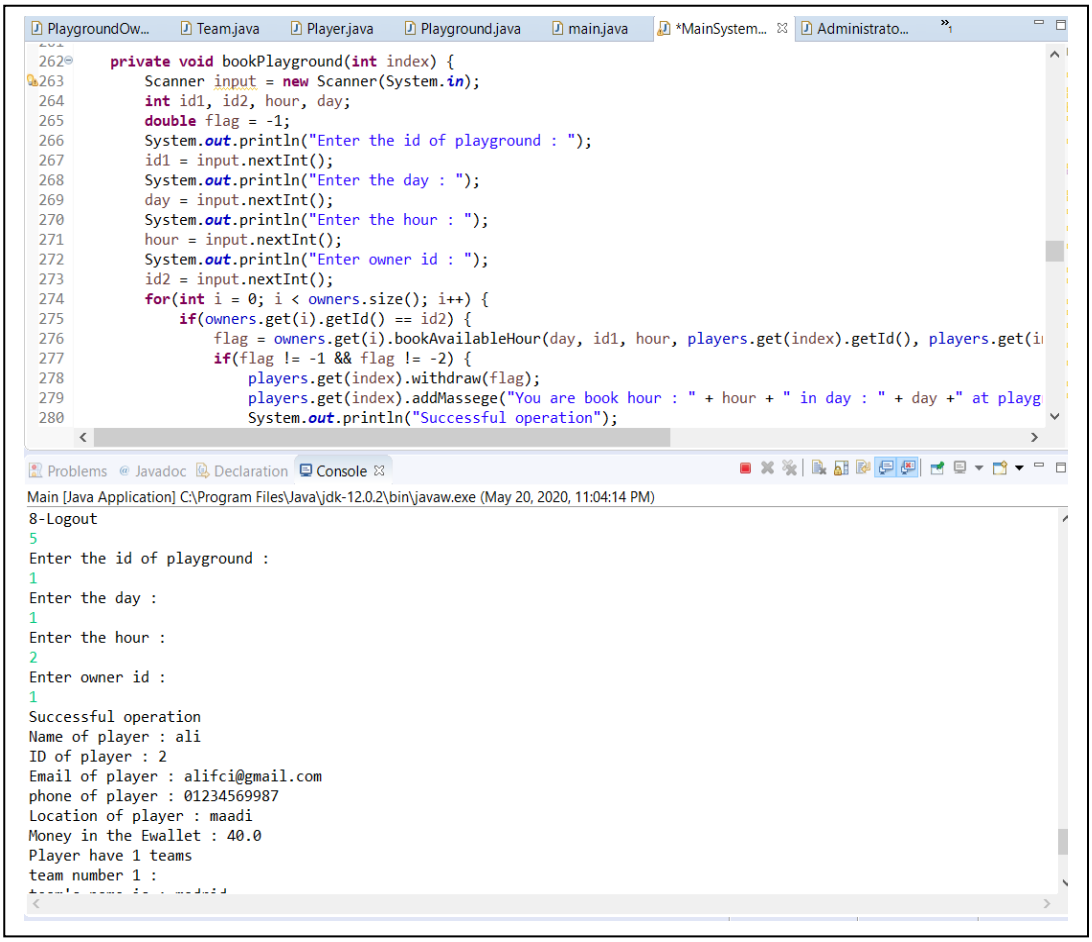
Problems Javadoc Declaration Console

Main [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (May 20, 2020, 11:04:14 PM)

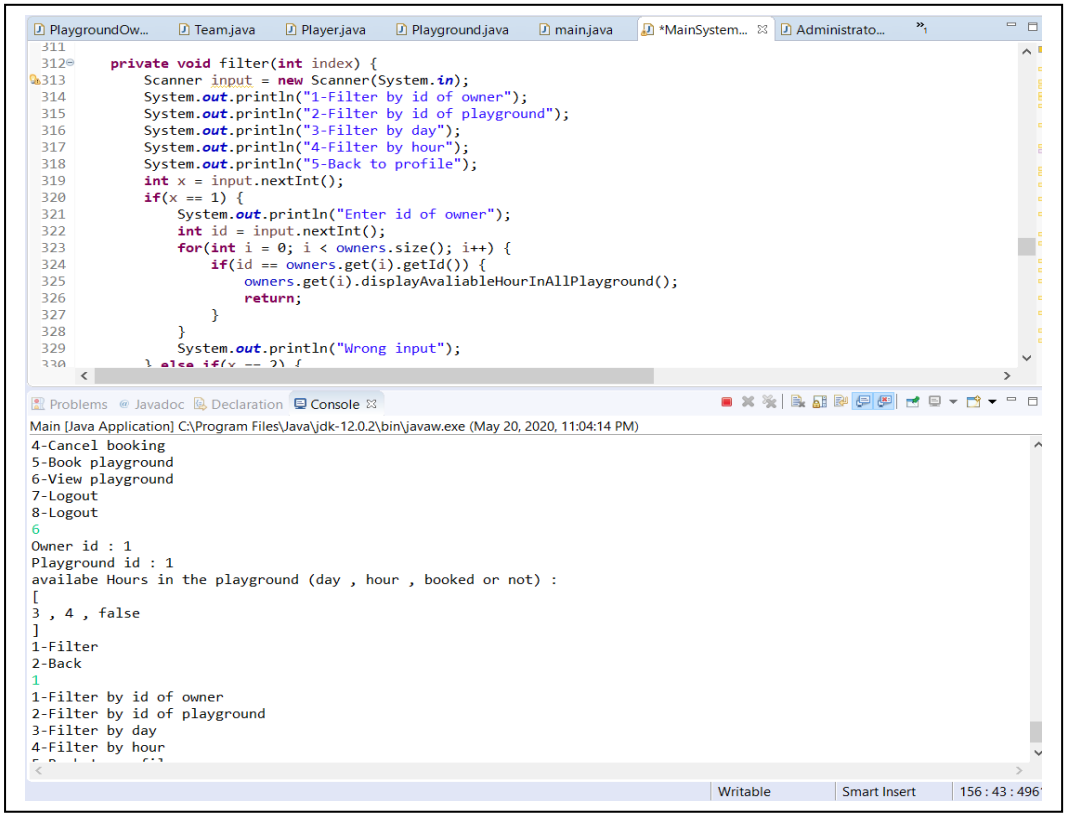
```
Enter location :
maadi
Enter role(player, owner) :
player
Name of player : ali
ID of player : 2
Email of player : alifci@gmail.com
phone of player : 01234569987
Location of player : maadi
Money in the Ewallet : 0.0
Player have 0 teams
1-Show all masseges
2-Create a team
3-Send invitation
4-Cancel booking
5-Book playground
6-View playground
```



```
PlaygroundOw... Team.java Player.java Playground.java main.java *MainSystem... Administrato...

184    private void createTeam(int index) {
185        Vector<String> playerEmails = new Vector<String>();
186        Scanner input = new Scanner(System.in);
187        String teamName = "", email = "", playerName = "";
188        System.out.println("Enter name of team : ");
189        teamName = input.nextLine();
190        System.out.println("Number if player (not more than 7 player) : ");
191        int x = input.nextInt();
192        Team t = new Team();
193        t.setName(teamName);
194        for(int i = 0; i < x; i++) {
195            Scanner in = new Scanner(System.in);
196            System.out.println("Enter name of player number " + (i + 1) + " : ");
197            playerName = in.nextLine();
198            System.out.println("Enter email of player number " + (i + 1) + " : ");
199            email = in.nextLine();
200            t.addplayer(playerName, email);
201        }
202        t.addplayer(players.get(index).getName(), players.get(index).getEmail());

Problems Javadoc Declaration Console
Main [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (May 20, 2020, 11:04:14 PM)
Enter email of player number 1 :
alaafci@gmail.com
Enter name of player number 2 :
hassan
Enter email of player number 2 :
hassanfci@gmail.com
Name of player : ali
ID of player : 2
Email of player : alifci@gmail.com
phone of player : 01234569987
Location of player : maadi
Money in the Ewallet : 0.0
Player have 1 teams
team number 1 :
team's name is : madrid
team's id is : 1
name of player is : alaa his email is : alaafci@gmail.com
name of player is : hassan his email is : hassanfci@gmail.com
name of player is : ali his email is : alifci@gmail.com
1-Show all masseges
```



```
PlaygroundOw... Team.java Player.java Playground.java main.java *MainSystem... Administrato...

262    private void bookPlayground(int index) {
263        Scanner input = new Scanner(System.in);
264        int id1, id2, hour, day;
265        double flag = -1;
266        System.out.println("Enter the id of playground : ");
267        id1 = input.nextInt();
268        System.out.println("Enter the day : ");
269        day = input.nextInt();
270        System.out.println("Enter the hour : ");
271        hour = input.nextInt();
272        System.out.println("Enter owner id : ");
273        id2 = input.nextInt();
274        for(int i = 0; i < owners.size(); i++) {
275            if(owners.get(i).getId() == id2) {
276                flag = owners.get(i).bookAvailableHour(day, id1, hour, players.get(index).getId(), players.get(i
277                if(flag != -1 && flag != -2) {
278                    players.get(index).withdraw(flag);
279                    players.get(index).addMassege("You are book hour : " + hour + " in day : " + day +" at playg
280                    System.out.println("Successful operation");

Problems Javadoc Declaration Console
Main [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (May 20, 2020, 11:04:14 PM)
8-Logout
5
Enter the id of playground :
1
Enter the day :
1
Enter the hour :
2
Enter owner id :
1
Successful operation
Name of player : ali
ID of player : 2
Email of player : alifci@gmail.com
phone of player : 01234569987
Location of player : maadi
Money in the Ewallet : 40.0
Player have 1 teams
team number 1 :
```



- **Link for GitHub repo :-**

  https://github.com/muhmedhaniiii/Gofo

- **Link for google drive :-**

  https://drive.google.com/drive/folders/1oXVI4KkA3pxCLpZcYkJXOYeWLSs3uwYz?usp=sharing

Code Listing

1. Administrator class

```
package project;

import java.util.Vector;

public class Administrator extends User {
	public Vector<String> masseges = new Vector<String>();

	public Administrator() {
		super();
	};

	public Administrator(String name, String password, String email, String
phone, String location, String role) {
		super(name, password, email, phone, location, role);
	}

	public void addmassge(String massege) {
		masseges.add(massege);
	}

	public void displaymasseges() {
		if (masseges.isEmpty()) {
			System.out.println("there is no message");
		} else {
			for (int i = masseges.size() - 1, j = 1; i > -1; i--, j++) {
				System.out.println("massege " + j + " : " +
masseges.get(i));
			}
		}
	}

	public void displayInfo() {
		System.out.println(
				"Name : " + getName() + " ,Email : " + getEmail() + "
,Phone : " + getPhone() + " ,ID : " + getId());
	}

}
```

2. Player class

```
package project;

import java.util.Vector;

/**
 * @author kerols
 */
public class Player extends User {
	/**
	 * Attribute to save all objects of team
	 */
	private Vector<Team> teams = new Vector<Team>();
	/**
	 * Attribute to save all email
	 */
	private Vector<String> masseges = new Vector<String>();
	/**
	 * Attribute to save the money which player owned
	 */
	private double Ewallet;

	/**
	 * Constructor to set attribute and set Ewallet to zero
	 *
	 * @param name
	 * @param password
	 * @param email
	 * @param phone
	 * @param location
	 * @param role
	 */
	public Player(String name, String password, String email, String phone,
String location, String role) {
		super(name, password, email, phone, location, role);
		Ewallet = 0;
	}

	/**
	 * Default constructor and set Ewallet to zero
	 */
	public Player() {
		super();
		Ewallet = 0;
	}

	/**
	 * Make player enter amount of money to the Ewallet
```

```
 *
 * @param amountMoney
 */
public void deposit(double amountMoney) {
	Ewallet += amountMoney;
};

/**
 * Make player get amount of money from this Ewallet
 *
 * @param amountMoney
 * @return boolean
 */
public boolean withdraw(double amountMoney) {
	if (amountMoney > Ewallet) {
		return false;
	}
	Ewallet -= amountMoney;
	return true;
};

/**
 * Make player know how much money in this Ewallet?
 *
 * @return double
 */
public double getEwallet() {
	return Ewallet;
};

/**
 * Display all data of player
 */
public void displayInformation() {
	System.out.println("Name of player : " + super.getName());
	System.out.println("ID of player : " + super.getId());
	System.out.println("Email of player : " + super.getEmail());
	System.out.println("phone of player : " + super.getPhone());
	System.out.println("Location of player : " + super.getLocation());
	System.out.println("Money in the Ewallet : " + Ewallet);
	System.out.println("Player have " + teams.size() + " teams");
	for (int i = 0; i < teams.size(); i++) {
		System.out.println("team number " + (i + 1) + " : ");
		teams.get(i).dispaly();
	}
}

/**
 * Add object of team to his vector
```

```
	 *
	 * @param t
	 */
	public void addTeam(Team t) {
		for (int i = 0; i < teams.size(); i++) {
			if (teams.get(i).getId() == t.getId()) {
				return;
			}
		}
		teams.add(t);
	}

	/**
	 * Display all email of player
	 */
	public void showAllMasseges() {
		if (masseges.isEmpty()) {
			System.out.println("there is no message");
		} else {
			for (int i = masseges.size() - 1, j = 1; i > -1; i--, j++) {
				System.out.println("massege " + j + " : " +
masseges.get(i));
			}
		}
	}

	/**
	 * Add email to his vector
	 *
	 * @param massege
	 */
	public void addMassege(String massege) {
		masseges.add(massege);
	}

	/**
	 * Add email to his vector if the id of team is one of team id which user
have it
	 *
	 * @param massege
	 * @param id
	 * @return boolean
	 */
	public boolean addMassege(String massege, int id) {
		for (int i = 0; i < teams.size(); i++) {
			if (teams.get(i).getId() == id) {
				masseges.add(massege);
				return true;
			}
```

```
            }
            return false;
      }
}
```

3. Playground class

```
package project;

/**
 *
 * @author Mohamed Saeed
 *
 */
public class Playground {
      /**
       * Attributes to save name, size, owner name, location, period of
cancellation
       */
      private String name, size, ownerName, location, cancellitionPeriod;
      /**
       * Attributes of price per hour and id of playground
       */
      private int price_hour, id;
      /**
       * Attribute to generate id without repeat
       */
      private static int ID = 1;
      /**
       * Attribute to save if playground active or not
       */
      private boolean activation;
      /**
       * Attribute to save all available hours
       */
      private boolean availableHours[][] = new boolean[30][24];
      /**
       * Attribute to save all booked hours
       */
      private boolean bookedHours[][] = new boolean[30][24];

      /**
       * Getter method to name
       *
       * @return String
       */
```

```
public String getName() {
	return name;
}

/**
 * Setter method to name
 *
 * @param name
 */
public void setName(String name) {
	this.name = name;
}

/**
 * Getter method to size
 *
 * @return String
 */
public String getSize() {
	return size;
}

/**
 * Setter method to size
 *
 * @param size
 */
public void setSize(String size) {
	this.size = size;
}

/**
 * Getter method to owner name
 *
 * @return String
 */
public String getOwnerName() {
	return ownerName;
}

/**
 * Setter method to owner name
 *
 * @param ownerName
 */
public void setOwnerName(String ownerName) {
	this.ownerName = ownerName;
}
```

```
/**
 * Getter method to location
 *
 * @return String
 */
public String getLocation() {
	return location;
}

/**
 * Setter method to location
 *
 * @param location
 */
public void setLocation(String location) {
	this.location = location;
}

/**
 * Getter method to period of cancellation
 *
 * @return String
 */
public String getCancellitionPeriod() {
	return cancellitionPeriod;
}

/**
 * Setter method to period of cancellation
 *
 * @param cancellitionPeriod
 */
public void setCancellitionPeriod(String cancellitionPeriod) {
	this.cancellitionPeriod = cancellitionPeriod;
}

/**
 * Getter method to price per hour
 *
 * @return integer
 */
public int getPrice_hour() {
	return price_hour;
}

/**
 * Setter method to price per hour
 *
 * @param price_hour
```

```
	 */
	public void setPrice_hour(int price_hour) {
		this.price_hour = price_hour;
	}

	/**
	 * Getter method to id
	 *
	 * @return integer
	 */
	public int getId() {
		return id;
	};

	/**
	 * Getter method to activation of playground ( active or not )
	 *
	 * @return
	 */
	public boolean isActivation() {
		return activation;
	}

	/**
	 * Setter method to activation of playground ( active or not ) by
administrator
	 *
	 * @param admin
	 * @return boolean
	 */
	public boolean setActivation(Administrator admin) {
		if (admin.getEmail().equalsIgnoreCase("XXXXX@gmail.com") &&
admin.getId() == 0) {
			activation = true;
			System.out.println("Successful operation");
			return true;
		}
		return false;
	};

	/**
	 * Add available hour
	 *
	 * @param day
	 * @param hour
	 */
	public void addAvailabeHours(int day, int hour) {
		availableHours[day - 1][hour] = true;
	}
```

```
	/**
	 * Setter method to Booked Hours
	 *
	 * @param day
	 * @param hour
	 * @param flag
	 * @return boolean
	 */
	public boolean setBookedHours(int day, int hour, boolean flag) {
		if (activation && availableHours[day - 1][hour]) {
			bookedHours[day - 1][hour] = flag;
			return true;
		}
		return false;
	}

	/**
	 * Default constructor
	 */
	public Playground() {
		activation = false;
		id = ID;
		ID++;
	}

	/**
	 * Constructor to set attribute
	 *
	 * @param name
	 * @param size
	 * @param ownerName
	 * @param location
	 * @param cancellitionPeriod
	 * @param price_hour
	 */
	public Playground(String name, String size, String ownerName, String
location, String cancellitionPeriod,
			int price_hour) {
		this.name = name;
		this.size = size;
		this.ownerName = ownerName;
		this.location = location;
		this.cancellitionPeriod = cancellitionPeriod;
		this.price_hour = price_hour;
		this.id = ID;
		ID++;
		activation = false;
	}
```

```
	/**
	 * Display all data of playground
	 *
	 * @param id
	 */
	public void displayPlayground(int id) {
		if (activation == true || id == 0) {
			System.out.println("ID of the playground : " + this.id);
			System.out.println("Name of the playground : " + name);
			System.out.println("Size of playground : " + size);
			System.out.println("name of the owner : " + ownerName);
			System.out.println("Location of the playground : " +
location);
			System.out.println("CancellitionPeriod of the playground : " +
cancellitionPeriod);
			System.out.println("Price of one hour : " + price_hour);
			System.out.println("availabe Hours in the playground (day ,
hour , booked or not) : ");
			System.out.println("[ ");
			for (int i = 0; i < 30; i++) {
				for (int j = 0; j < 24; j++) {
					if (availableHours[i][j]) {
						System.out.println(i + 1 + " , " + j + " , "
+ bookedHours[i][j]);
					}
				}
			}
			System.out.println("]");
		} else {
			System.out.println("Playground with id : " + id + " is not
active");
		}
	};

	/**
	 * Display all available hours
	 */
	public void displayAvaliableHours() {
		if (activation) {
			System.out.println("Playground id : " + id);
			System.out.println("availabe Hours in the playground (day ,
hour , booked or not) : ");
			System.out.println("[ ");
			for (int i = 0; i < 30; i++) {
				for (int j = 0; j < 24; j++) {
					if (availableHours[i][j] && !bookedHours[i][j]) {
						System.out.println(i + 1 + " , " + j + " , "
+ bookedHours[i][j]);
```

```
                        }
                    }
                }
                System.out.println("]");
            } else {
                System.out.println("Playground with id : " + id + " is not
active");
            }
        };

        /**
         * Display available hours with this day and hour
         *
         * @param from
         * @param to
         * @param choose
         */
        public void displayAvaliableHours(int from, int to, String choose) {
            if (activation) {
                System.out.println("Playground id : " + id);
                int start1 = 0, start2 = 0, end1 = 30, end2 = 24;
                if (choose.equalsIgnoreCase("day")) {
                    start1 = from;
                    end1 = to;
                } else {
                    start1 = from;
                    end1 = to;
                }
                System.out.println("availabe Hours in the playground : ");
                System.out.println("[ ");
                for (int i = start1; i < end1; i++) {
                    for (int j = start2; j < end2; j++) {
                        if (availableHours[i][j] && !bookedHours[i][j]) {
                            System.out.println("Day : " + i + 1 + " ,
Hour : " + j);
                        }
                    }
                }
                System.out.println("]");
            } else {
                System.out.println("Playground with id : " + id + " is not
active");
            }};
}
```

4. Playground Owner class

```
package project;

import java.util.Vector;

/**
 *
 * @author Omnia
 *
 */
public class PlaygroundOwner extends User {
	/**
	 * Attribute to save all objects of playground
	 */
	private Vector<Playground> Playgrounds = new Vector<Playground>();
	/**
	 * Attribute to save all email
	 */
	private Vector<String> masseges = new Vector<String>();
	/**
	 * Attribute to save the money which owner owned
	 */
	private double Ewallet;

	/**
	 * Default constructor and set Ewallet to zero
	 */
	public PlaygroundOwner() {
		super();
		Ewallet = 0;
	};

	/**
	 * Constructor to set attribute and set Ewallet to zero
	 *
	 * @param name
	 * @param password
	 * @param email
	 * @param phone
	 * @param location
	 * @param role
	 */
	public PlaygroundOwner(String name, String password, String email, String
phone, String location, String role) {
		super(name, password, email, phone, location, role);
		Ewallet = 0;
	}
```

```
	/**
	 * Display all data of owner
	 */
	public void createProfile() {
		System.out.println("Name : " + getName() + " ,Email : " + getEmail()
+ " ,Phone : " + getPhone() + " ,ID : "
				+ getId() + " ,Ewallet : " + Ewallet);
		if (Playgrounds.size() == 0) {
			System.out.println("Owner has not any playgrounds yet");
		} else {
			System.out.println("Playgrounds owned by this owner " +
Playgrounds.size() + " : ");
			for (int i = 0; i < Playgrounds.size(); i++) {
				Playgrounds.get(i).displayPlayground(1);
			}
		}
	}

	/**
	 * Add object of playground to his vector
	 *
	 * @param obj
	 */
	public void addPlayground(Playground obj) {
		Playgrounds.add(obj);
	}

	/**
	 * Display all email of player
	 */
	public void showAllMasseges() {
		if (masseges.isEmpty()) {
			System.out.println("there is no message");
		} else {
			for (int i = masseges.size() - 1, j = 1; i > -1; i--, j++) {
				System.out.println("massege " + j + " : " +
masseges.get(i));
			}
		}
	}

	/**
	 * Add email to his vector
	 *
	 * @param massege
	 */
	public void addMassege(String massege) {
		masseges.add(massege);
	}
```

```
	/**
	 * Book available hour in playground with this id and check if use have
enough money
	 *
	 * @param day
	 * @param id
	 * @param hour
	 * @param idOfPlayer
	 * @param money
	 * @return double
	 */
	public double bookAvailableHour(int day, int id, int hour, int idOfPlayer,
double money) {
		boolean flag;
		for (int i = 0; i < Playgrounds.size(); i++) {
			if (Playgrounds.get(i).getId() == id) {
				if (money < Playgrounds.get(i).getPrice_hour()) {
					return -2;
				}
				flag = Playgrounds.get(i).setBookedHours(day, hour,
true);
				if (flag) {
					deposit(Playgrounds.get(i).getPrice_hour());
					addMassege("playground of id : " + id + " has
booked at day : " + day + " at hour : " + hour
							+ " by player with id : " +
idOfPlayer);
					return (Playgrounds.get(i).getPrice_hour());
				}
				break;
			}
		}
		return -1;
	};

	/**
	 * Display available hour in all playground owned by owner
	 */
	public void displayAvaliableHourInAllPlayground() {
		System.out.println("Owner id : " + super.getId());
		for (int i = 0; i < Playgrounds.size(); i++) {
			Playgrounds.get(i).displayAvaliableHours();
		}
	};

	/**
	 * Display available hour in Playground with this id if it exists
	 *
```

```
	 * @param id
	 * @return boolean
	 */
	public boolean displayAvaliableHourInSpecialPlayground(int id) {
		for (int i = 0; i < Playgrounds.size(); i++) {
			if (id == Playgrounds.get(i).getId()) {
				System.out.println("Owner id : " + super.getId());
				Playgrounds.get(i).displayAvaliableHours();
				return true;
			}
		}
		return false;
	};

	/**
	 * Display available hour in all playground at special hour or day owned
by owner
	 *
	 * @param start
	 * @param end
	 * @param choose
	 */
	public void displayAvaliableHourInAllPlaygroundAtSpecialHourOrDay(int
start, int end, String choose) {
		System.out.println("Owner id : " + super.getId());
		for (int i = 0; i < Playgrounds.size(); i++) {
			Playgrounds.get(i).displayAvaliableHours(start, end, choose);
		}
	};

	/**
	 * Activate playground by send object of administrator and check if it
correct id
	 *
	 * @param admin
	 * @param id
	 * @return boolean
	 */
	public boolean activateplayground(Administrator admin, int id) {
		for (int i = 0; i < Playgrounds.size(); i++) {
			if (Playgrounds.get(i).getId() == id) {
				return Playgrounds.get(i).setActivation(admin);
			}
		}
		return false;
	};

	/**
	 * Display playground with this id
```

```
	 *
	 * @param id
	 * @return boolean
	 */
	public boolean displayPlayground(int id) {
		for (int i = 0; i < Playgrounds.size(); i++) {
			if (Playgrounds.get(i).getId() == id) {
				Playgrounds.get(i).displayPlayground(0);
				return true;
			}
		}
		return false;
	};

	/**
	 * Make player enter amount of money to the Ewallet
	 *
	 * @param amountMoney
	 */
	public void deposit(double amountMoney) {
		Ewallet += amountMoney;
	};

	/**
	 * Make player get amount of money from this Ewallet
	 *
	 * @param amountMoney
	 * @return boolean
	 */
	public boolean withdraw(double amountMoney) {
		if (amountMoney > Ewallet) {
			return false;
		}
		Ewallet -= amountMoney;
		return true;
	};

	/**
	 * Make player know how much money in this Ewallet?
	 *
	 * @return double
	 */
	public double getEwallet() {
		return Ewallet;
	};
}
```

5. Team class

```
package project;

import java.util.Vector;

/**
 *
 * @author Khaled
 *
 */
public class Team {
	/**
	 * Attribute to save name of team
	 */
	private String name;
	/**
	 * Attribute to save id of team
	 */
	private int id;
	/**
	 * Attribute to generate id without repeat
	 */
	private static int ID = 1;
	/**
	 * Attribute to save all players name
	 */
	private Vector<String> Playername = new Vector<String>();
	/**
	 * Attribute to save all players email
	 */
	private Vector<String> PlayerEmail = new Vector<String>();

	/**
	 * Default constructor
	 */
	public Team() {
		id = ID;
		ID++;
	};

	/**
	 * Constructor to set attribute
	 *
	 * @param name
	 */
	public Team(String name) {
		this.name = name;
		id = ID;
```

```
            ID++;
      }

      /**
       * Getter method to name
       *
       * @return String
       */
      public String getName() {
            return name;
      }

      /**
       * Setter method to name
       *
       * @param name
       */
      public void setName(String name) {
            this.name = name;
      }

      /**
       * Getter method to id
       *
       * @return integer
       */
      public int getId() {
            return id;
      }

      /**
       * Display all data of team
       */
      public void dispaly() {
            System.out.println("team's name is : " + name);
            System.out.println("team's id is : " + id);
            for (int i = 0; i < Playername.size(); i++) {
                  System.out.println("name of player is : " + Playername.get(i)
+ " his email is : " + PlayerEmail.get(i));
            }
      }

      /**
       * To add player ( name , email )
       *
       * @param name
       * @param email
       */
      public void addplayer(String name, String email) {
```

```
            for (int i = 0; i < PlayerEmail.size(); i++) {
                  if (PlayerEmail.get(i) == email) {
                        return;
                  }
            }
            Playername.add(name);
            PlayerEmail.add(email);
      }

      /**
       * To get vector which save all players email
       *
       * @return Vector<String>
       */
      public Vector<String> getEmailsOfPlayers() {
            return PlayerEmail;
      };
   }
```

6. Main class

```
package project;

/**
 *
 * @author kerols , Khaled , Mohamed Hany , Mohamed Saeed , Omnia
 * @version 1.0
 * @since 2020-5-21
 */
public class main {

      public static void main(String[] args) {
            MainSystem m = new MainSystem();
            m.mainMethod();
      }

}
```

7. User class

```
package project;

/**
 *
 * @author kerols
 *
 */
public class User {
	/**
	 * Attribute to generate id without repeat
	 */
	private static int ID = 0;
	/**
	 * Attributes to save name , password , email , phone , location , role of
user
	 */
	private String name, password, email, phone, location, role;
	/**
	 * Attribute to save id
	 */
	private int id;

	/**
	 * Default constructor
	 */
	public User() {
		this.id = ID;
		ID++;
	};

	/**
	 * Constructor to set attribute
	 *
	 * @param name
	 * @param password
	 * @param email
	 * @param phone
	 * @param location
	 * @param role
	 */
	public User(String name, String password, String email, String phone,
String location, String role) {
		super();
		this.name = name;
		this.password = password;
		this.email = email;
		this.phone = phone;
```

```
        this.location = location;
        this.role = role;
        this.id = ID;
        ID++;
};

/**
 * Getter method to name
 *
 * @return String
 */
public String getName() {
        return name;
};

/**
 * Setter method to name
 *
 * @param name
 */
public void setName(String name) {
        this.name = name;
};

/**
 * Getter method to password
 *
 * @return String
 */
public String getPassword() {
        return password;
};

/**
 * Setter method to password
 *
 * @param password
 */
public void setPassword(String password) {
        this.password = password;
};

/**
 * Getter method to email
 *
 * @return String
 */
public String getEmail() {
        return email;
```

```
};

/**
 * Setter method to email
 *
 * @param email
 */
public void setEmail(String email) {
	this.email = email;
};

/**
 * Getter method to phone
 *
 * @return String
 */
public String getPhone() {
	return phone;
};

/**
 * Setter method to phone
 *
 * @param phone
 */
public void setPhone(String phone) {
	this.phone = phone;
};

/**
 * Getter method to location
 *
 * @return String
 */
public String getLocation() {
	return location;
};

/**
 * Setter method to location
 *
 * @param location
 */
public void setLocation(String location) {
	this.location = location;
};

/**
 * Getter method to role
```

```
     *
     * @return String
     */
    public String getRole() {
          return role;
    };

    /**
     * Setter method to role
     *
     * @param role
     */
    public void setRole(String role) {
          this.role = role;
    };

    /**
     * Getter method to id
     *
     * @return integer
     */
    public int getId() {
          return id;
    };

    /**
     * Setter method to id
     *
     * @param id
     */
    public void setId(int id) {
          this.id = id;
    };
}
```

8. Main system class

```
package project;

import java.util.Scanner;
import java.util.Vector;

/**
 * @author kerols
 */
public class MainSystem {
	/**
	 * Attribute vector of player object to save all objects of player
	 */
	private Vector<Player> players = new Vector<Player>();
	/**
	 * Attribute vector of owner object to save all objects of owner
	 */
	private Vector<PlaygroundOwner> owners = new Vector<PlaygroundOwner>();
	/**
	 * Default attribute of administrator
	 */
	private Administrator admin = new Administrator("ahmed", "123456",
"XXXXX@gmail.com", "01234567890", "6-october",
			"administrator");

	/**
	 * Main method which control every thing Display menu to make user choose
( login or register or exit )
	 */
	public void mainMethod() {
		while (true) {
			Scanner input = new Scanner(System.in);
			System.out.println("1-Login");
			System.out.println("2-Register");
			System.out.println("3-Exit");
			int x;
			x = input.nextInt();
			if (x == 1) {
				login();
			} else if (x == 2) {
				register();
			} else if (x == 3) {
				System.out.println("Thank you for using this program");
				break;
			} else {
				System.out.println("Wrong input");
			}
		}
```

```
			return;
		};

		/**
		 * Method to create object from class user ( player or owner ) Make user
	enter data
		 */
		private void register() {
			Scanner input = new Scanner(System.in);
			String name, password1, password2, email, phone, location, role;
			System.out.println("Enter name : ");
			name = input.nextLine();
			System.out.println("Enter Password : ");
			password1 = input.nextLine();
			System.out.println("Re-enter Password : ");
			password2 = input.nextLine();
			while (true) {
				Scanner in = new Scanner(System.in);
				if (password1.equalsIgnoreCase(password2)) {
					break;
				}
				System.out.println("The two password not equal");
				System.out.println("Enter Password : ");
				password1 = in.nextLine();
				System.out.println("Re-enter Password : ");
				password2 = in.nextLine();
			}
			System.out.println("Enter Email : ");
			email = input.nextLine();
			while (!checkEmail(email)) {
				Scanner in = new Scanner(System.in);
				System.out.println("Please enter correct Email");
				email = in.nextLine();
			}
			System.out.println("Enter phone : ");
			phone = input.nextLine();
			while (true) {
				Scanner in = new Scanner(System.in);
				if (phone.length() == 11) {
					break;
				}
				System.out.println("Please enter correct phone number");
				phone = in.nextLine();
			}
			System.out.println("Enter location : ");
			location = input.nextLine();
			System.out.println("Enter role(player, owner) : ");
			role = input.nextLine();
			while (true) {
```

```
                Scanner in = new Scanner(System.in);
                if (role.equalsIgnoreCase("player")) {
                        Player object = new Player(name, password1, email,
phone, location, role);
                        players.add(object);
                        profileOfPlayer(players.size() - 1);
                        break;
                } else if (role.equalsIgnoreCase("owner")) {
                        PlaygroundOwner object = new PlaygroundOwner(name,
password1, email, phone, location, role);
                        owners.add(object);
                        profileOfowner(owners.size() - 1);
                        break;
                }
                System.out.println("Wrong input");
                System.out.println("Enter player or owner : ");
                role = in.nextLine();
        }
    };

    /**
     * Method to check email is correct or not
     *
     * @param email
     * @return boolean
     */
    private boolean checkEmail(String email) {
        boolean x = false, y = false;
        for (int i = 0; i < email.length(); i++) {
            if (email.charAt(i) == '.') {
                x = true;
            } else if (email.charAt(i) == '@') {
                y = true;
            }
        }
        if (x && y) {
            return true;
        }
        return false;
    };

    /**
     * Method make user enter data and check if this data saved in system or
not
     *
     * To make user login in system
     */
    private void login() {
        Scanner input = new Scanner(System.in);
```

```
            String email, password, role;
            System.out.println("Enter Email : ");
            email = input.nextLine();
            System.out.println("Enter Password : ");
            password = input.nextLine();
            System.out.println("Enter Role(player, owner, administrator) : ");
            role = input.nextLine();
            if (role.equalsIgnoreCase("administrator")) {
                  if (admin.getEmail().equalsIgnoreCase(email) &&
admin.getPassword().equalsIgnoreCase(password)
                              && admin.getRole().equalsIgnoreCase(role)) {
                        profileOfadmin();
                        return;
                  } else {
                        System.out.println("invalid email or password or role");
                  }
            } else if (role.equalsIgnoreCase("player")) {
                  for (int i = 0; i < players.size(); i++) {
                        if (players.get(i).getEmail().equalsIgnoreCase(email)
                                    &&
players.get(i).getPassword().equalsIgnoreCase(password)
                                    &&
players.get(i).getRole().equalsIgnoreCase(role)) {
                              profileOfPlayer(i);
                              return;
                        }
                  }
            } else if (role.equalsIgnoreCase("owner")) {
                  for (int i = 0; i < owners.size(); i++) {
                        if (owners.get(i).getEmail().equalsIgnoreCase(email)
                                    &&
owners.get(i).getPassword().equalsIgnoreCase(password)
                                    &&
owners.get(i).getRole().equalsIgnoreCase(role)) {
                              profileOfowner(i);
                              return;
                        }
                  }
            }
            System.out.println("invalid email or password or role");
            return;
      };

      /**
       * Method to display all data of player and small menu with all player
options
       *
       * Attribute index is position of object player in vector players
       *
```

```
         * @param index
         */
        private void profileOfPlayer(int index) {
              while (true) {
                    Scanner input = new Scanner(System.in);
                    players.get(index).displayInformation();
                    System.out.println("1-Show all masseges");
                    System.out.println("2-Create a team");
                    System.out.println("3-Send invitation");
                    System.out.println("4-Cancel booking");
                    System.out.println("5-Book playground");
                    System.out.println("6-View playground");
                    System.out.println("7-deposit money to Ewallet");
                    System.out.println("8-Logout");
                    int x = input.nextInt();
                    if (x == 1) {
                          players.get(index).showAllMasseges();
                    } else if (x == 2) {
                          createTeam(index);
                    } else if (x == 3) {
                          sendInvitation();
                    } else if (x == 4) {
                          System.out.println("this option has not been added
yet");
                    } else if (x == 5) {
                          bookPlayground(index);
                    } else if (x == 6) {
                          viewPlayground(index);
                    } else if (x == 7) {
                          System.out.println("Enter the amount of money : ");
                          double money;
                          money = input.nextDouble();
                          players.get(index).deposit(money);
                    } else if (x == 8) {
                          break;
                    } else {
                          System.out.println("Wrong input");
                    }
              }
        };

        /**
         * Method to make user enter data of team and create object of team and
add it in all players object in the team
         *
         * Attribute index is position of object player in vector players
         *
         * @param index
         */
```

```
	private void createTeam(int index) {
		Vector<String> playerEmails = new Vector<String>();
		Scanner input = new Scanner(System.in);
		String teamName = "", email = "", playerName = "";
		System.out.println("Enter name of team : ");
		teamName = input.nextLine();
		System.out.println("Number if player (not more than 7 player) : ");
		int x = input.nextInt();
		Team t = new Team();
		t.setName(teamName);
		for (int i = 0; i < x; i++) {
			Scanner in = new Scanner(System.in);
			System.out.println("Enter name of player number " + (i + 1) +
" : ");
			playerName = in.nextLine();
			System.out.println("Enter email of player number " + (i + 1) +
" : ");
			email = in.nextLine();
			t.addplayer(playerName, email);
		}
		t.addplayer(players.get(index).getName(),
players.get(index).getEmail());
		playerEmails = t.getEmailsOfPlayers();
		for (int i = 0; i < players.size(); i++) {
			for (int j = 0; j < playerEmails.size(); j++) {
				if
(playerEmails.get(j).equalsIgnoreCase(players.get(i).getEmail())) {
					players.get(i).addTeam(t);
				}
			}
		}
	};

	/**
	 * Make user send message to another player or team
	 */
	private void sendInvitation() {
		Scanner input = new Scanner(System.in);
		String mss, email;
		int id;
		System.out.println("Emter the massege : ");
		mss = input.nextLine();
		System.out.println("You want to send massege to\n 1-team \n 2-
player");
		int x = input.nextInt();
		if (x == 1) {
			System.out.println("Enter id of team : ");
			id = input.nextInt();
			boolean flag, flag2 = true;
```

```
                for (int i = 0; i < players.size(); i++) {
                        flag = players.get(i).addMassege(mss, id);
                        if (flag) {
                                flag2 = false;
                        }
                }
                if (flag2) {
                        System.out.println("Wrong input");
                }
        } else if (x == 2) {
                String y;
                while (true) {
                        Scanner in = new Scanner(System.in);
                        System.out.println("Enter email of player : ");
                        email = in.nextLine();
                        boolean flag = true;
                        for (int i = 0; i < players.size(); i++) {
                                if
(players.get(i).getEmail().equalsIgnoreCase(email)) {
                                        players.get(i).addMassege(mss);
                                        flag = false;
                                        break;
                                }
                        }
                        if (flag) {
                                System.out.println("Wrong input");
                        }
                        System.out.println("You want to send email to another
player( y , n )");
                        y = in.nextLine();
                        if (y.equalsIgnoreCase("n")) {
                                break;
                        }
                }
        } else {
                System.out.println("Wrong input");
        }
    };

    /**
     * Make user do a booking by enter the data of slot Attribute index is
position of object player in vector players
     *
     * @param index
     */
    private void bookPlayground(int index) {
        Scanner input = new Scanner(System.in);
        int id1, id2, hour, day;
        double flag = -1;
```

```
            System.out.println("Enter the id of playground : ");
            id1 = input.nextInt();
            System.out.println("Enter the day : ");
            day = input.nextInt();
            System.out.println("Enter the hour : ");
            hour = input.nextInt();
            System.out.println("Enter owner id : ");
            id2 = input.nextInt();
            for (int i = 0; i < owners.size(); i++) {
                  if (owners.get(i).getId() == id2) {
                        flag = owners.get(i).bookAvailableHour(day, id1, hour,
players.get(index).getId(),
                                    players.get(index).getEwallet());
                        if (flag != -1 && flag != -2) {
                              players.get(index).withdraw(flag);
                              players.get(index).addMassege("You are book hour :
" + hour + " in day : " + day
                                          + " at playground id : " + id1 + "
which owner id : " + id2);
                              System.out.println("Successful operation");
                              return;
                        }
                        break;
                  }
            }
            if (flag == -2) {
                  System.out.println("you do not have enough money");
                  return;
            }
            System.out.println("Wrong input");
      }

      /**
       * Display all available hours in all playground Display option to user to
filter it
       *
       * Attribute index is position of object player in vector players
       *
       * @param index
       */
      private void viewPlayground(int index) {
            for (int i = 0; i < owners.size(); i++) {
                  owners.get(i).displayAvaliableHourInAllPlayground();
            }
            while (true) {
                  Scanner input = new Scanner(System.in);
                  System.out.println("1-Filter");
                  System.out.println("2-Back");
                  int x = input.nextInt();
```

```
                if (x == 1) {
                        filter(index);
                } else if (x == 2) {
                        break;
                } else {
                        System.out.println("Wrong input");
                }
        }
};

/**
 * Make user filter available hours in playground
 *
 * Attribute index is position of object player in vector players
 *
 * @param index
 */
private void filter(int index) {
        Scanner input = new Scanner(System.in);
        System.out.println("1-Filter by id of owner");
        System.out.println("2-Filter by id of playground");
        System.out.println("3-Filter by day");
        System.out.println("4-Filter by hour");
        System.out.println("5-Back to profile");
        int x = input.nextInt();
        if (x == 1) {
                System.out.println("Enter id of owner");
                int id = input.nextInt();
                for (int i = 0; i < owners.size(); i++) {
                        if (id == owners.get(i).getId()) {

owners.get(i).displayAvaliableHourInAllPlayground();
                                return;
                        }
                }
                System.out.println("Wrong input");
        } else if (x == 2) {
                System.out.println("Enter id of playground");
                int id = input.nextInt();
                boolean flag;
                for (int i = 0; i < owners.size(); i++) {
                        flag =
owners.get(i).displayAvaliableHourInSpecialPlayground(id);
                        if (flag) {
                                return;
                        }
                }
                System.out.println("Wrong input");
        } else if (x == 3) {
```

```
                  int start, end;
                  while (true) {
                        Scanner in = new Scanner(System.in);
                        System.out.println("Enter two days from to in two
different line (min 1 : max 31) : ");
                        start = in.nextInt();
                        end = in.nextInt();
                        if (start > 0 && end < 32) {
                              break;
                        }
                        System.out.println("Wrong input");
                  }
                  for (int i = 0; i < owners.size(); i++) {

      owners.get(i).displayAvaliableHourInAllPlaygroundAtSpecialHourOrDay(start
- 1, end - 1, "day");
                  }
            } else if (x == 4) {
                  int start, end;
                  while (true) {
                        Scanner in = new Scanner(System.in);
                        System.out.println("Enter two hours from to in two
different line (min 0 : max 24) : ");
                        start = in.nextInt();
                        end = in.nextInt();
                        if (start > -1 && end < 25) {
                              break;
                        }
                        System.out.println("Wrong input");
                  }
                  for (int i = 0; i < owners.size(); i++) {

      owners.get(i).displayAvaliableHourInAllPlaygroundAtSpecialHourOrDay(start,
end, "hour");
                  }
            } else if (x != 5) {
                  System.out.println("Wrong input");
            }
      };

      /**
       * Method to display all data of administrator and small menu with all
administrator options
       */
      private void profileOfadmin() {
            while (true) {
                  Scanner input = new Scanner(System.in);
                  admin.displayInfo();
                  System.out.println("1-Show all masseges");
```

```
                System.out.println("2-Suspend");
                System.out.println("3-Delete");
                System.out.println("4-Activate");
                System.out.println("5-Aprove");
                System.out.println("6-Logout");
                int x = input.nextInt();
                if (x == 1) {
                      admin.displaymasseges();
                } else if (x == 2) {
                      System.out.println("this option has not been added
yet");
                } else if (x == 3) {
                      System.out.println("this option has not been added
yet");
                } else if (x == 4) {
                      System.out.println("this option has not been added
yet");
                } else if (x == 5) {
                      activatPlayground();
                } else if (x == 6) {
                      break;
                } else {
                      System.out.println("Wrong input");
                }
          }
    };

    /**
     * Make administrator approve playground
     */
    private void activatPlayground() {
          Scanner input = new Scanner(System.in);
          int id1, id2;
          boolean flag;
          System.out.println("Enter id of playground : ");
          id1 = input.nextInt();
          System.out.println("Enter id of owner : ");
          id2 = input.nextInt();
          for (int i = 0; i < owners.size(); i++) {
                if (owners.get(i).getId() == id2) {
                      System.out.println("Data of playground : ");
                      owners.get(i).displayPlayground(id1);
                      break;
                }
          }
          for (int i = 0; i < owners.size(); i++) {
                if (owners.get(i).getId() == id2) {
                      flag = owners.get(i).activateplayground(admin, id1);
                      if (flag) {
```

```
                                return;
                          }
                    }
              }
              System.out.println("Wrong input");
        };

        /**
         * Method to display all data of owner and small menu with all owner
options
         *
         * Attribute index is position of object owner in vector owners
         *
         * @param index
         */
        private void profileOfowner(int index) {
              while (true) {
                    Scanner input = new Scanner(System.in);
                    owners.get(index).createProfile();
                    System.out.println("1-Show all masseges");
                    System.out.println("2-Add playground");
                    System.out.println("3-Check ewllat");
                    System.out.println("4-Update playground");
                    System.out.println("5-View bookings");
                    System.out.println("6-Logout");
                    int x = input.nextInt();
                    if (x == 1) {
                          owners.get(index).showAllMasseges();
                    } else if (x == 2) {
                          addPlayground(index);
                    } else if (x == 3) {
                          System.out.print("Your money in Ewallet : ");
                          System.out.println(owners.get(index).getEwallet());
                    } else if (x == 4) {
                          System.out.println("this option has not been added
yet");
                    } else if (x == 5) {
                          System.out.println("this option has not been added
yet");
                    } else if (x == 6) {
                          break;
                    } else {
                          System.out.println("Wrong input");
                    }
              }
        };

        /**
         * Make owner add play ground by enter data of it
```

```
         *
         * Attribute index is position of object owner in vector owners
         *
         * @param index
         */
        private void addPlayground(int index) {
              Scanner input = new Scanner(System.in);
              String name, size, location, period;
              int price, number, h, d;
              System.out.println("Enter name of playground : ");
              name = input.nextLine();
              System.out.println("Enter size of playground (L : W) : ");
              size = input.nextLine();
              System.out.println("Enter location of playground : ");
              location = input.nextLine();
              System.out.println("Enter period of playground like(3d or 2h) : ");
              period = input.nextLine();
              System.out.println("Enter price per hour : ");
              price = input.nextInt();
              Playground ground = new Playground(name, size,
owners.get(index).getName(), location, period, price);
              System.out.println("Enter number of hours was available : ");
              number = input.nextInt();
              System.out.println("Enter day then hour in two different line (day
[1,30] hour [0,23] : ");
              for (int i = 0; i < number; i++) {
                    Scanner in = new Scanner(System.in);
                    d = in.nextInt();
                    h = in.nextInt();
                    if (d > 0 && d < 31 && h > -1 && h < 24) {
                          ground.addAvailabeHours(d, h);
                    }
              }
              owners.get(index).addPlayground(ground);
              admin.addmassge("Owner wit id : " + owners.get(index).getId() + "
make requst to add playground with id : "
                          + ground.getId());
        }

  }
```

Authors

- Mostafa Saad and Mohammad El-Ramly (Edited by Mohamed Samir) (V1.0)
- Updated by Mohammad El-Ramly (V2.0)