Cairo University  
Faculty of Computers and Information



**CS251**

**Software Engineering I**

GOFO

Software Design Specifications

Team

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Email** | **Mobile** |
| 20180218 | Mohamed Gamal Diab | Mohamed.gamal2052000@gmail.com | 01122485062 |
| 20180350 | Youssef Maher Maher | youssefmaher.yh@gmail.com | 01113453457 |
| 20180167 | Ali Mohamed Fathy | ali.mohamed.65666@gmail.com | 01153395569 |
| 20180051 | Alamir Hassan Younis | alamirhassan8@gmail.com | 01127171631 |

Contents

[Team 3](#_Toc42649968)

[Document Purpose and Audience 3](#_Toc42649969)

[System Models 4](#_Toc42649970)

[I. Class Diagram(s) 4](#_Toc42649971)

[II. Class Descriptions 5](#_Toc42649972)

[III. Sequence diagrams 6](#_Toc42649973)

[Class - Sequence Usage Table 12](#_Toc42649974)

[IV. User Interface Design 13](#_Toc42649975)

[Program implementation 20](#_Toc42649976)

[Tools 72](#_Toc42649977)

[Links 72](#_Toc42649978)

[Ownership Report 72](#_Toc42649979)

[References 73](#_Toc42649980)

[Authors 73](#_Toc42649981)

# Team

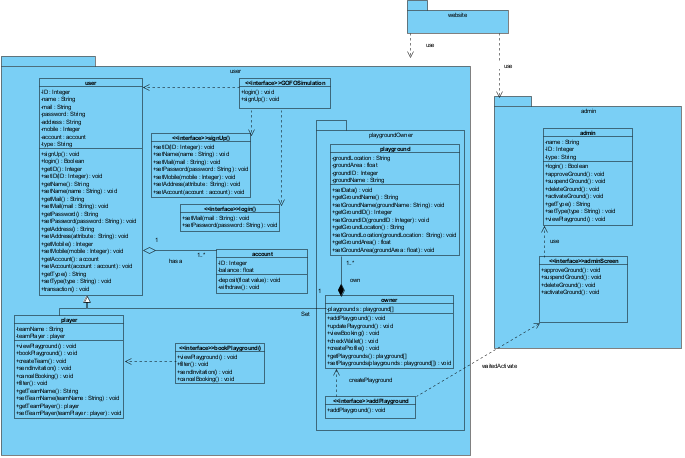
|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Email** | **Mobile** |
| 20180218 | Mohamed Gamal Diab | Mohamed.gamal2052000@gmail.com | 01122485062 |
| 20180350 | Youssef Maher Maher | youssefmaher.yh@gmail.com | 01113453457 |
| 20180167 | Ali Mohamed Fathy | ali.mohamed.65666@gmail.com | 01153395569 |
| 20180051 | Alamir Hassan Younis | alamirhassan8@gmail.com | 01127171631 |

# Document Purpose and Audience

* This document includes SRS description for GoFo football playground booking system. It describes the functions of the system and the uses cases.
* 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

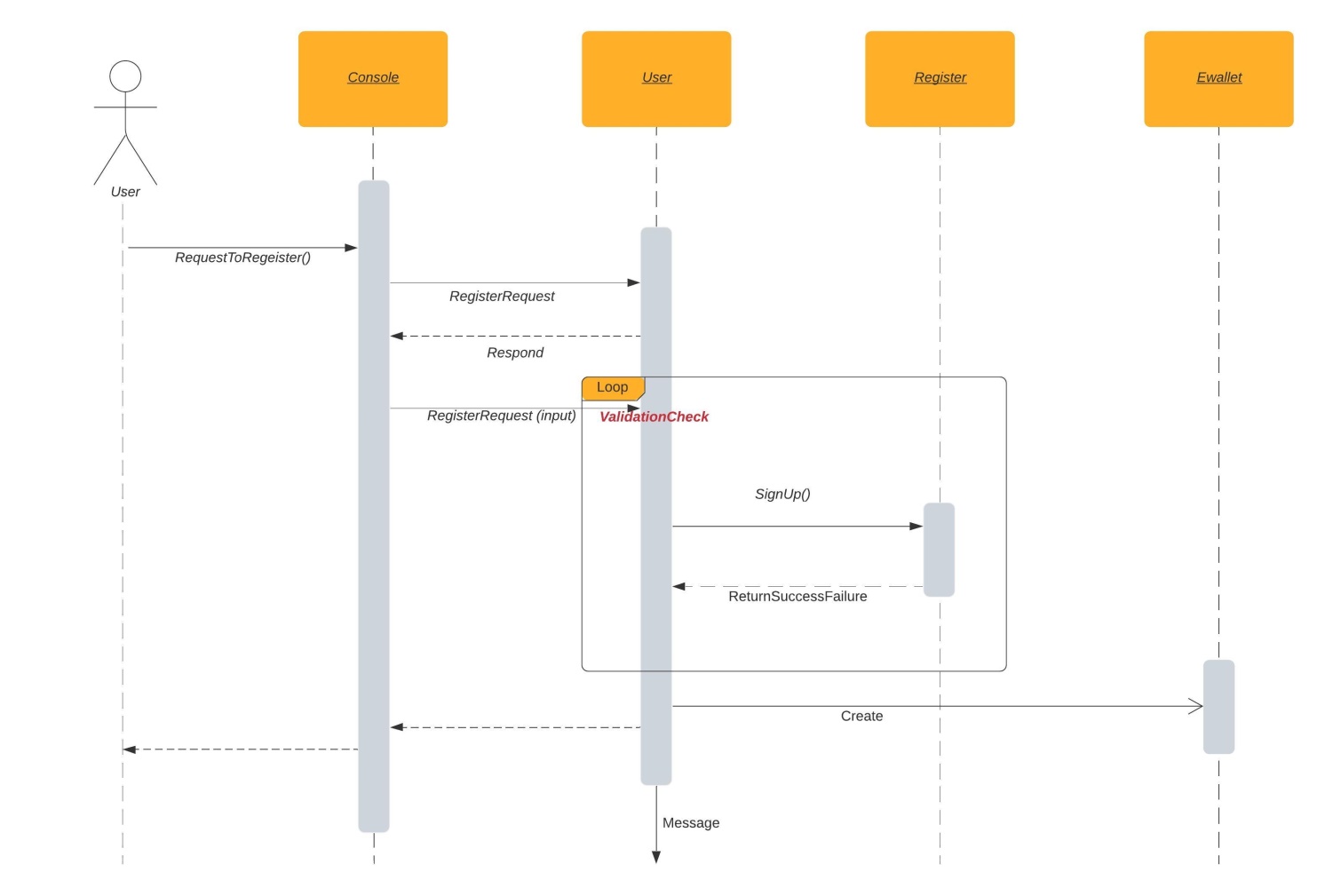
## I. Class Diagram(s)

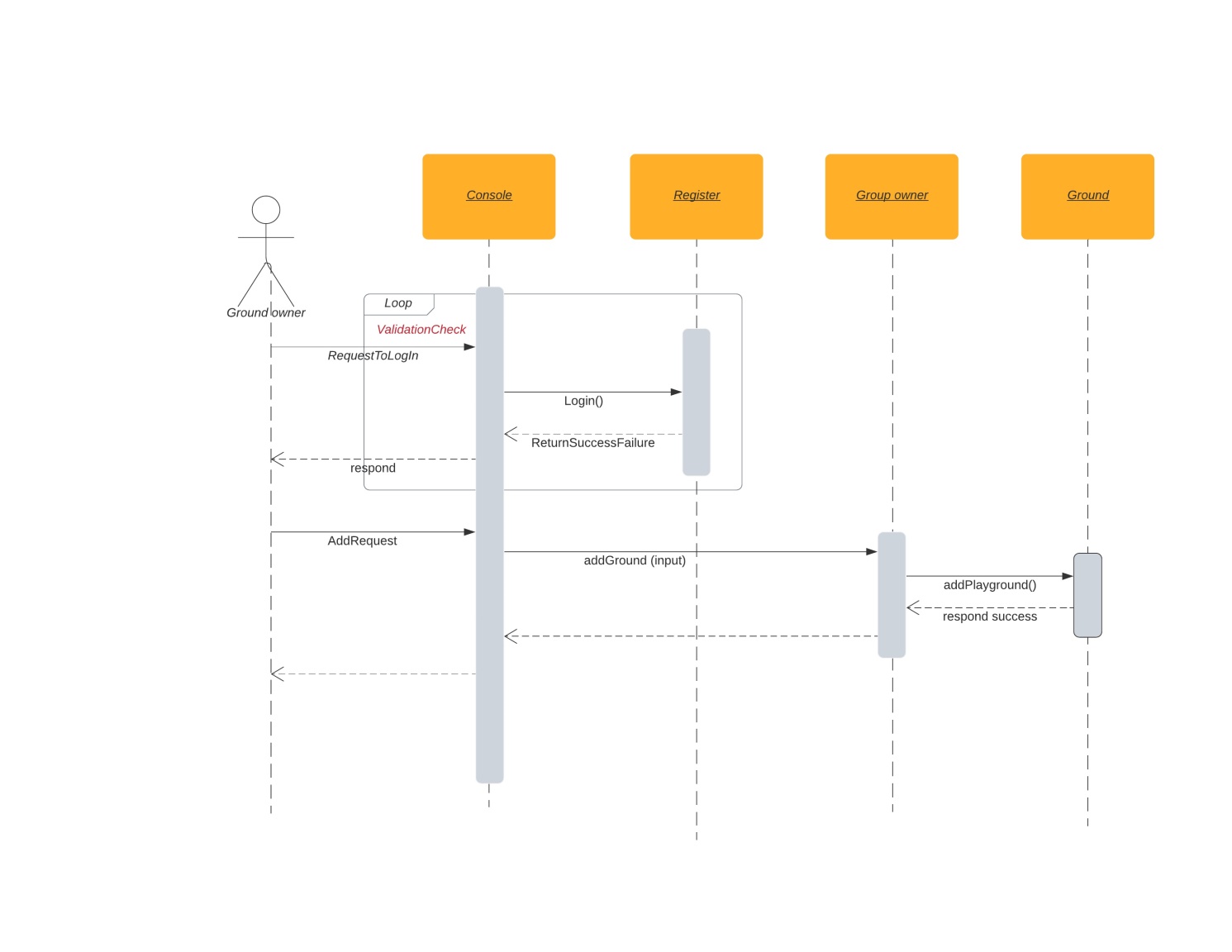


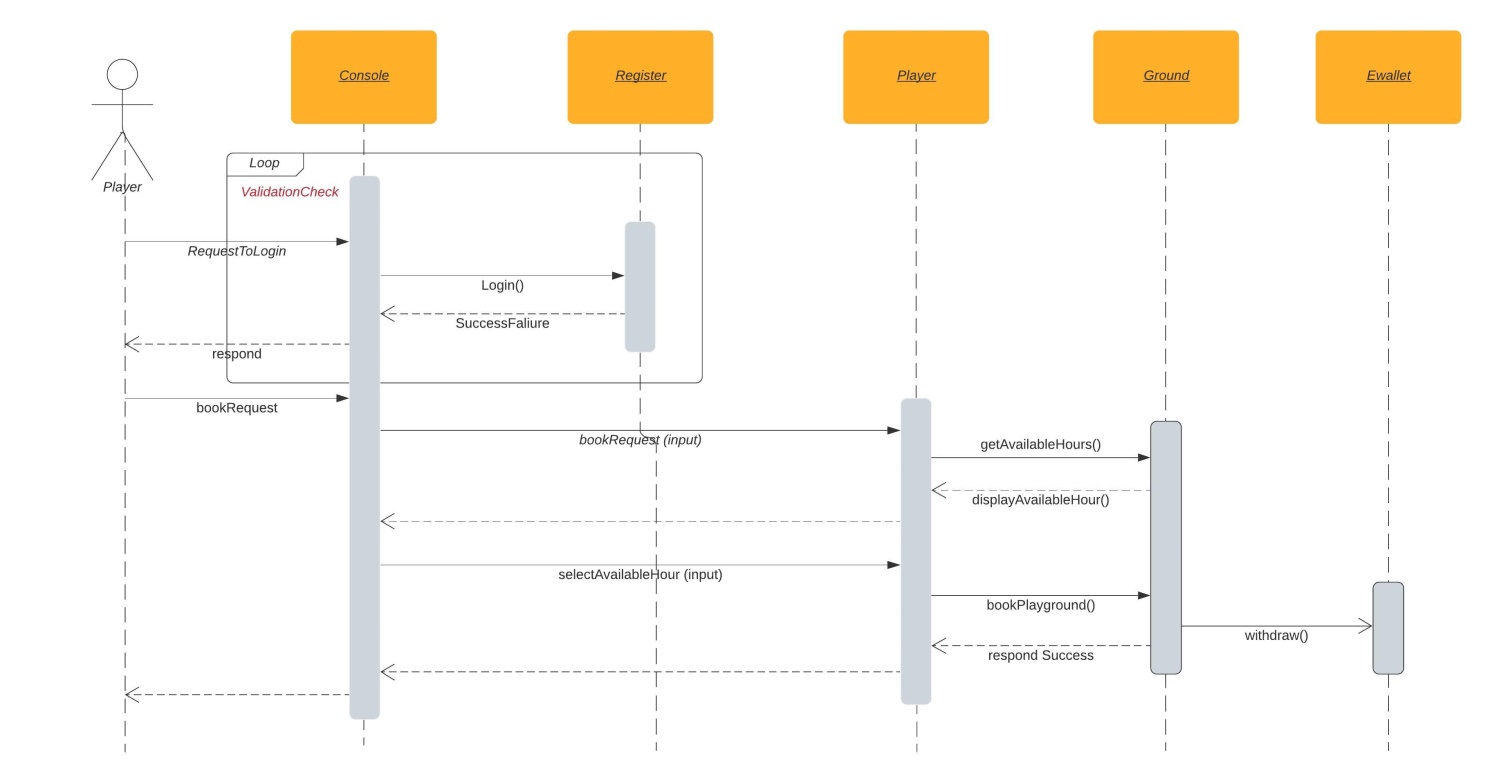
## II. Class Descriptions

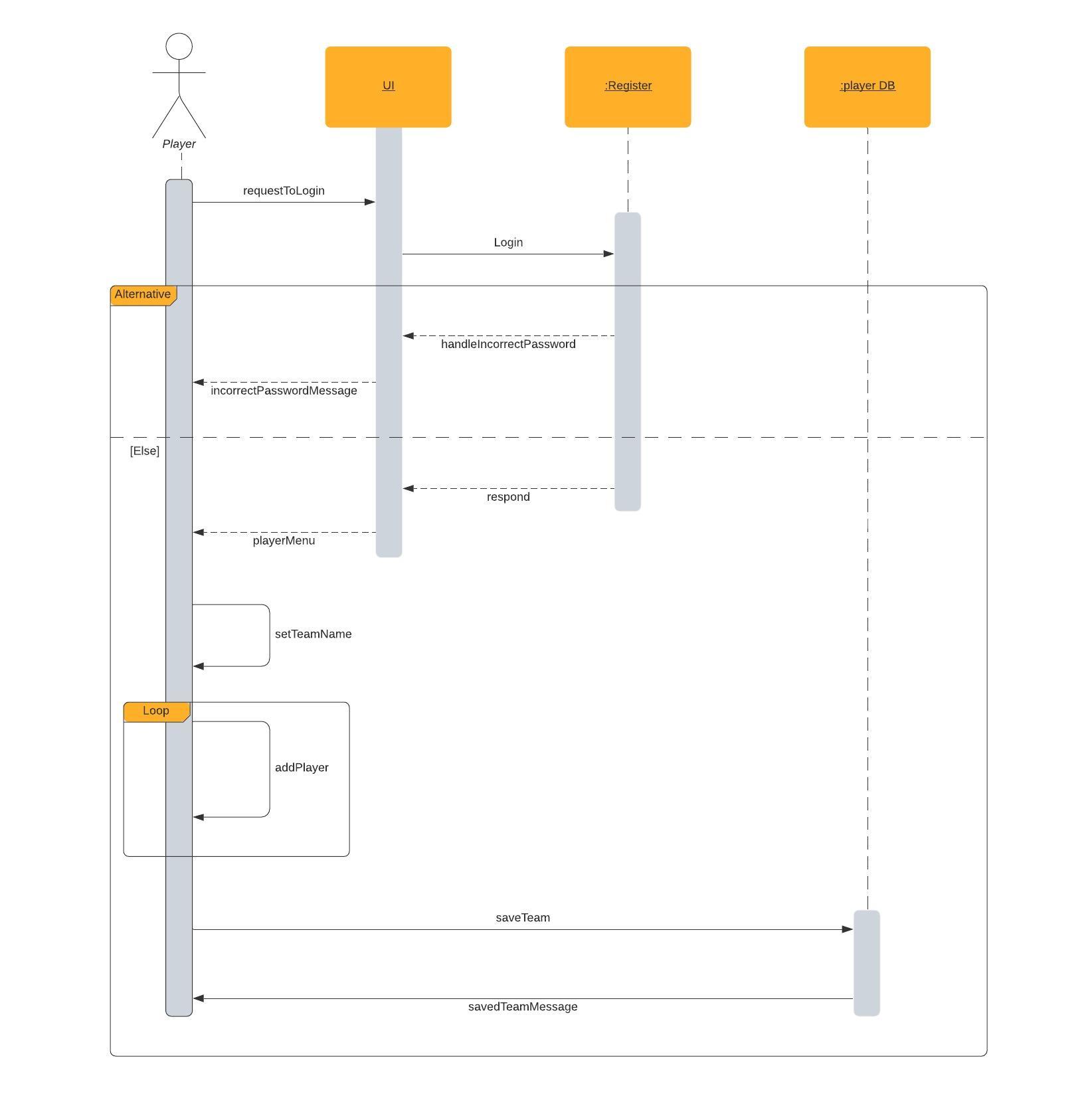
| **Class ID** | **Class Name** | **Description & Responsibility** |
| --- | --- | --- |
| 1. C1 | User | User is parent of two classes’ owner and player.  This class has 8 attributes (name, mail, password, ID, type, phone, address, account)  , have setter and getter for this attribute. |
| 2. C2 | Player | Player is child of class user. This class for player who wanted book playground. Player has 2 attributes name of team and player of this team.  Class has setter & getter of this attributes, and has functions (view Playground, book playground, filter playground, cancel booking, create team, view book). |
| 3. C3 | Owner | Owner is child of class user. This class for playground owner. Player has 1 attribute list of owner’s playground.  Class has setter & getter of this attributes, and has 5 functions (add playground, update playground, view booking, check ewallet,,view active playground, view playground). |
| 4. C4 | playground | Playground has 4 attributes (groundName, groundLocation, groundArea, groundID, state, availableHour, price) this attribute has setter and getter.  This class is important for owner it has all data about playground of owner display available hour booking. |
| 5. C5 | Ewallet | Every user should have account in ewallet to make transaction between player and owner. This class has 2 attributes (ID, balance), and 2 function (withdraw, deposit). |
| 6. C6 | admin | * + This class for admin he response on reliability between owner and player by functions [activatedGrounds](file:///D:\FCI_CU\FCAI-CU%202.2\project\SW\02.%20GoFo%20Implementation%20(TD)\doc\GoFoBackage\Admin.html#activatedGrounds(GoFoBackage.Register)), [activateGround](file:///D:\FCI_CU\FCAI-CU%202.2\project\SW\02.%20GoFo%20Implementation%20(TD)\doc\GoFoBackage\Admin.html#activateGround(GoFoBackage.Register,int)), [approveGround](file:///D:\FCI_CU\FCAI-CU%202.2\project\SW\02.%20GoFo%20Implementation%20(TD)\doc\GoFoBackage\Admin.html#approveGround(GoFoBackage.Register,int)), [deleteGround](file:///D:\FCI_CU\FCAI-CU%202.2\project\SW\02.%20GoFo%20Implementation%20(TD)\doc\GoFoBackage\Admin.html#deleteGround(GoFoBackage.Register,int)), [getAdminNAme](file:///D:\FCI_CU\FCAI-CU%202.2\project\SW\02.%20GoFo%20Implementation%20(TD)\doc\GoFoBackage\Admin.html#getAdminNAme()), [login](file:///D:\FCI_CU\FCAI-CU%202.2\project\SW\02.%20GoFo%20Implementation%20(TD)\doc\GoFoBackage\Admin.html#login(java.lang.String,java.lang.String)), [nonActivatedGrounds](file:///D:\FCI_CU\FCAI-CU%202.2\project\SW\02.%20GoFo%20Implementation%20(TD)\doc\GoFoBackage\Admin.html#nonActivatedGrounds(GoFoBackage.Register)), [getPassword](file:///D:\FCI_CU\FCAI-CU%202.2\project\SW\02.%20GoFo%20Implementation%20(TD)\doc\GoFoBackage\Admin.html#getPassword()), [setAdminName](file:///D:\FCI_CU\FCAI-CU%202.2\project\SW\02.%20GoFo%20Implementation%20(TD)\doc\GoFoBackage\Admin.html#setAdminName(java.lang.String)), [setPassword](file:///D:\FCI_CU\FCAI-CU%202.2\project\SW\02.%20GoFo%20Implementation%20(TD)\doc\GoFoBackage\Admin.html#setPassword(java.lang.String)), [suspendedGrounds](file:///D:\FCI_CU\FCAI-CU%202.2\project\SW\02.%20GoFo%20Implementation%20(TD)\doc\GoFoBackage\Admin.html#suspendedGrounds(GoFoBackage.Register)), [suspendGround](file:///D:\FCI_CU\FCAI-CU%202.2\project\SW\02.%20GoFo%20Implementation%20(TD)\doc\GoFoBackage\Admin.html#suspendGround(GoFoBackage.Register,int)) |
| 7.C7 | Register | This class like database contain this attribute (owner, player, ID, admin)  And two function sign up,login. |
| 8. C8 | GOFO simulation | This class is interface for user has two functions. Login for user registed before, signup to create account on GOFO. |
| 9. C9 | login | This class is interface for user. User enter his mail and password and system check and return true or false. |
| 10. C10 | Signup | This class is interface for user. He talks the data from user and the system create his mail. |
| 11.C11 | Book playground | The player use it to view playgrounds, filtered it, book it and send invitation to his team or other player and make transaction. |

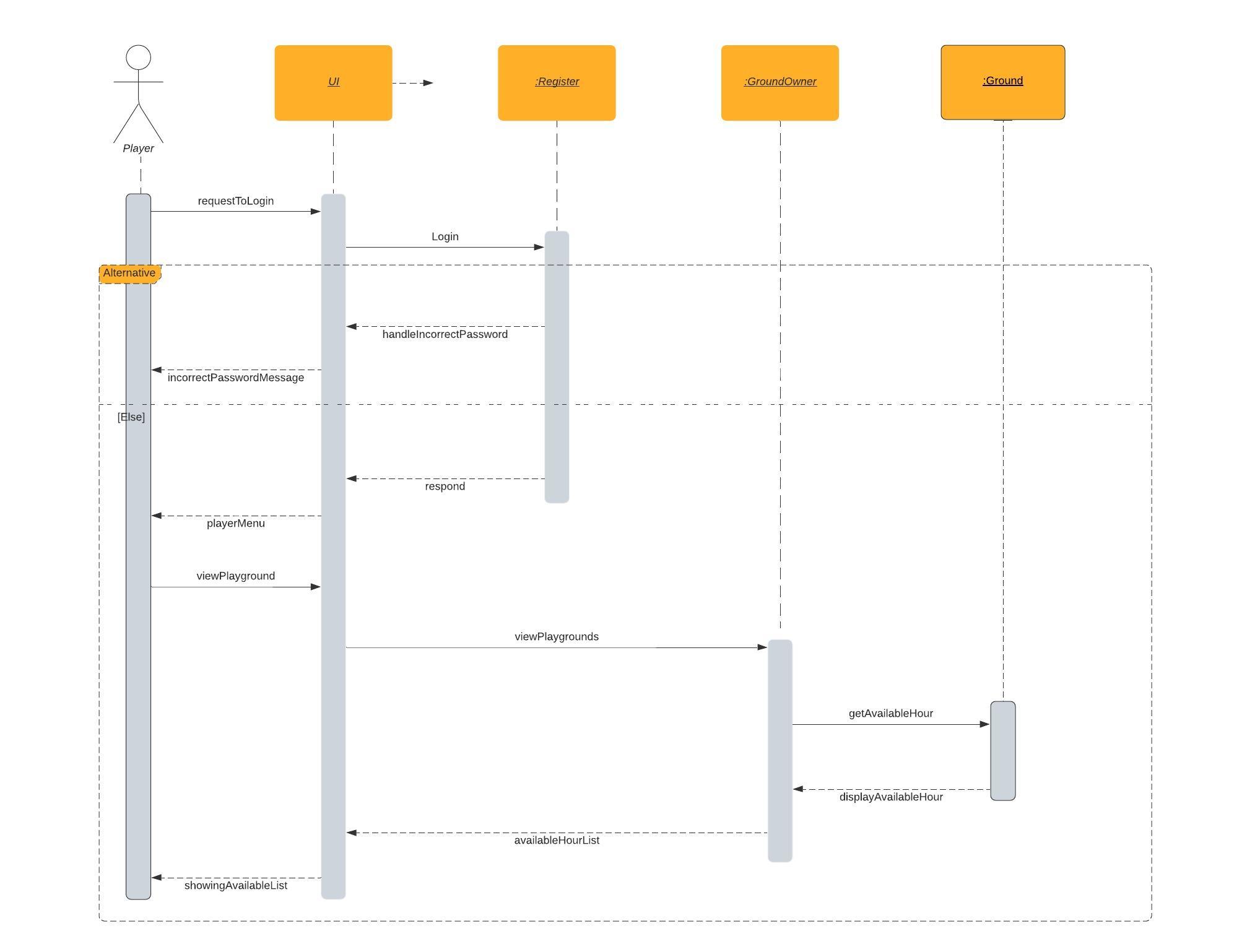
## III. Sequence diagrams

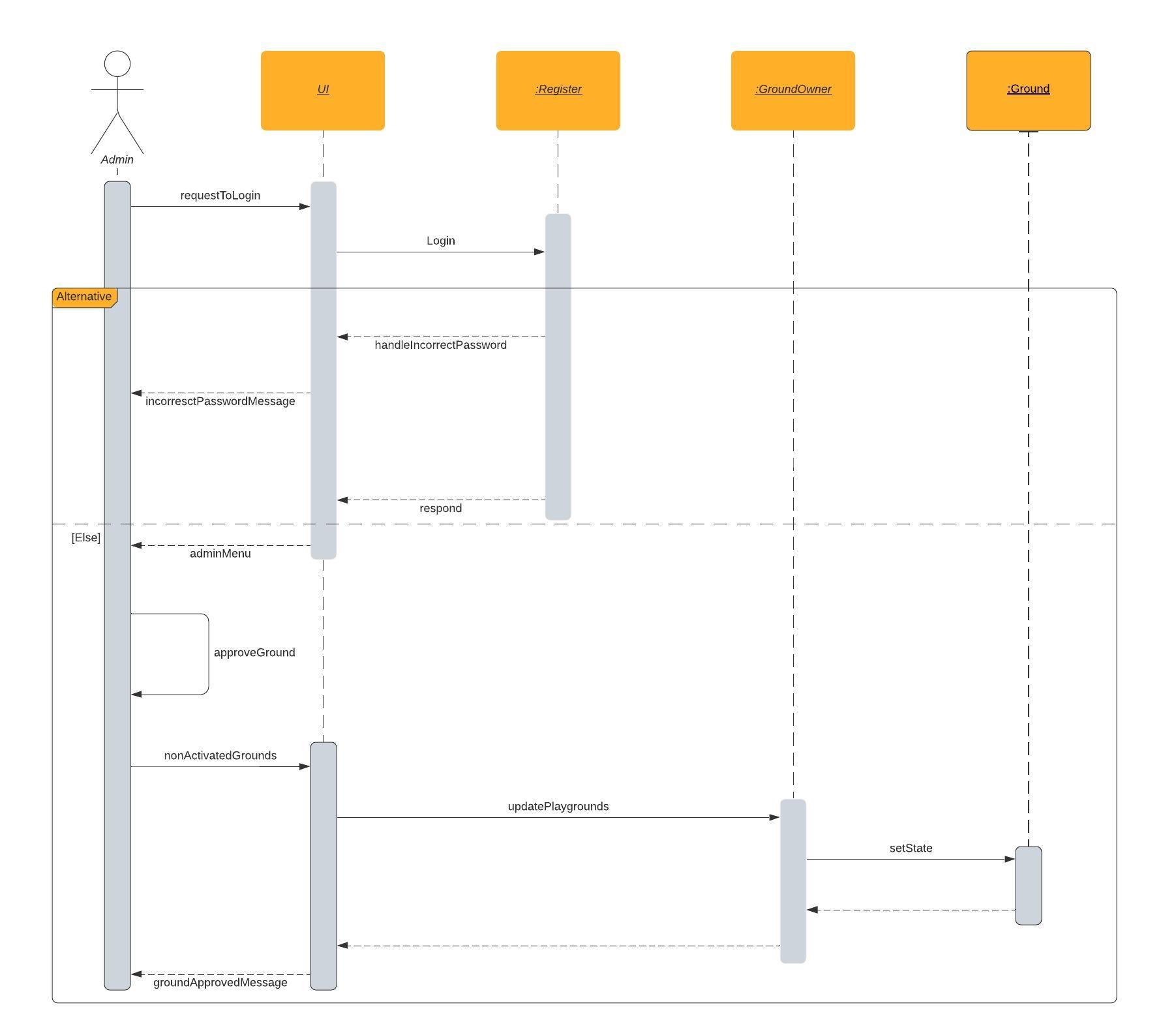
****











### 

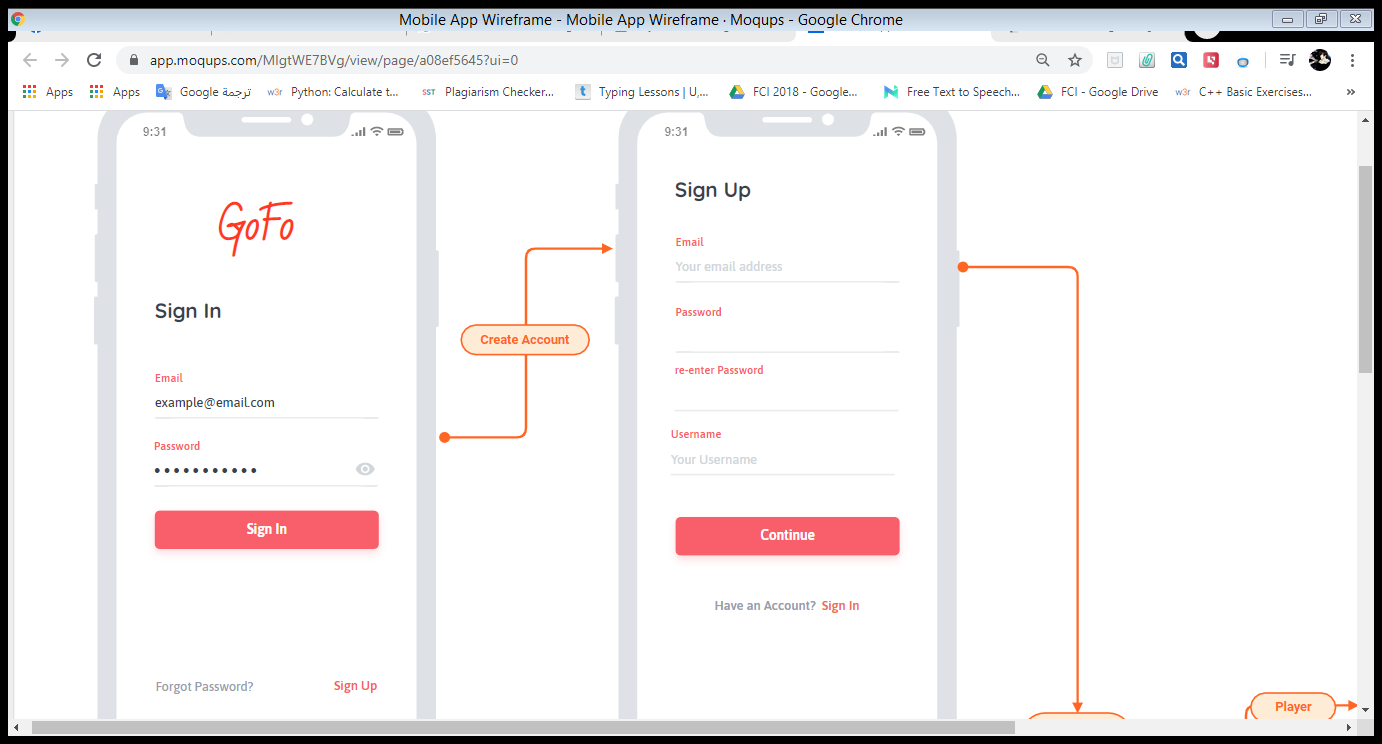
### Class - Sequence Usage Table

| **Sequence Diagram** | **Classes Used** | **All Methods Used** |
| --- | --- | --- |
| 1. Sign up | Class user  Class Player  Class owner | 1. signUp(String name, String mail, String password, String address, String mobile, String type) |
| 1. Add Playground | Class owner | 1. addPlayground(String name, String location, **float** Area, **int** price) |
| 1. Book playground | Class Player | 1. bookPlayground(Register users, **int** ownerID, **int** first, **int** last, **int** groundID) 2. withdraw(**int** valueWithdraw) 3. booking(**int** ID, **int** first, **int** last, **int** cost, **int** groundID) 4. checkbooking(**int** first, **int** last, **int** groundID) |
| 1. Create team | Class Player | 1. addPlayer(String playerName, Register users) |
| 1. view playground hour | Class Player | 1. viewPlayground(Register users) 2. viewPlaygrounds() |
| 1. Approve playground | Class Admin | 1. nonActivatedGrounds(Register user) 2. approveGround(Register user, **int** id) 3. setState(String state) |

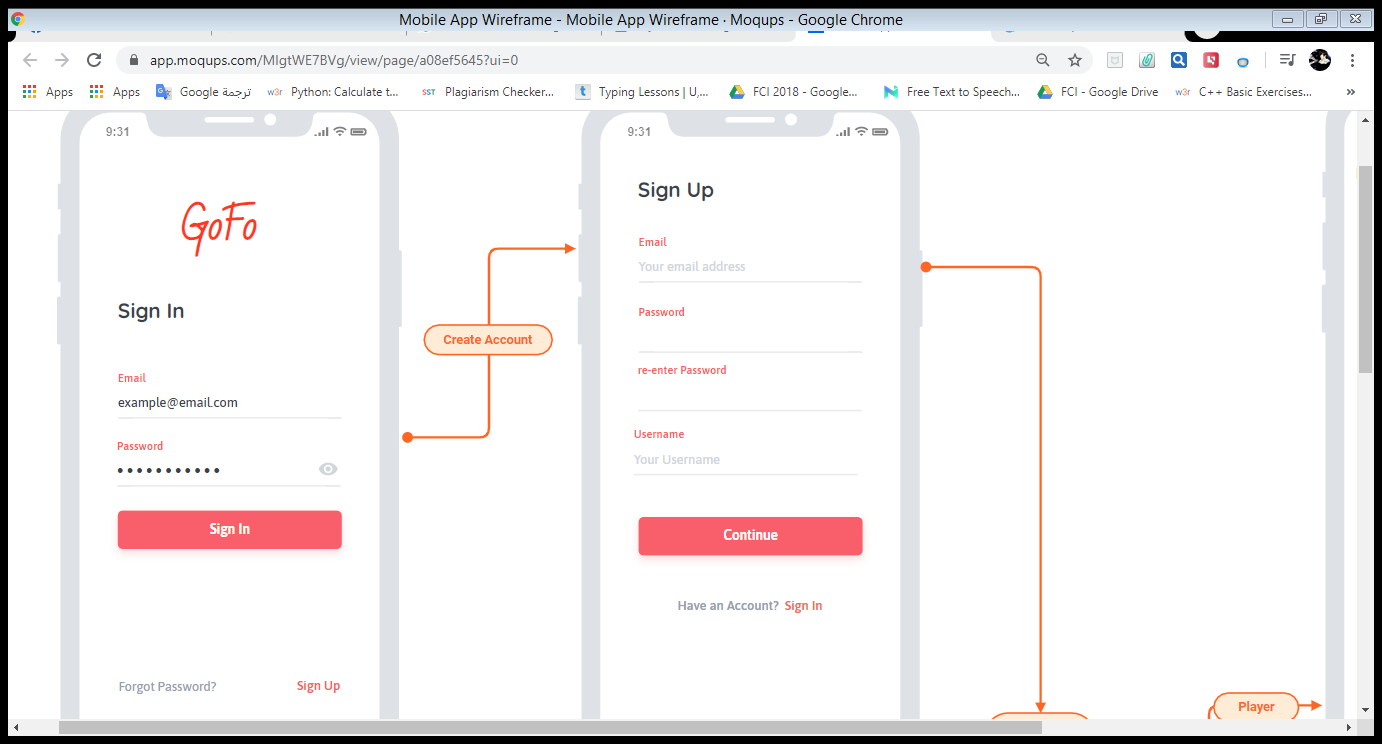
## 

## IV. User Interface Design

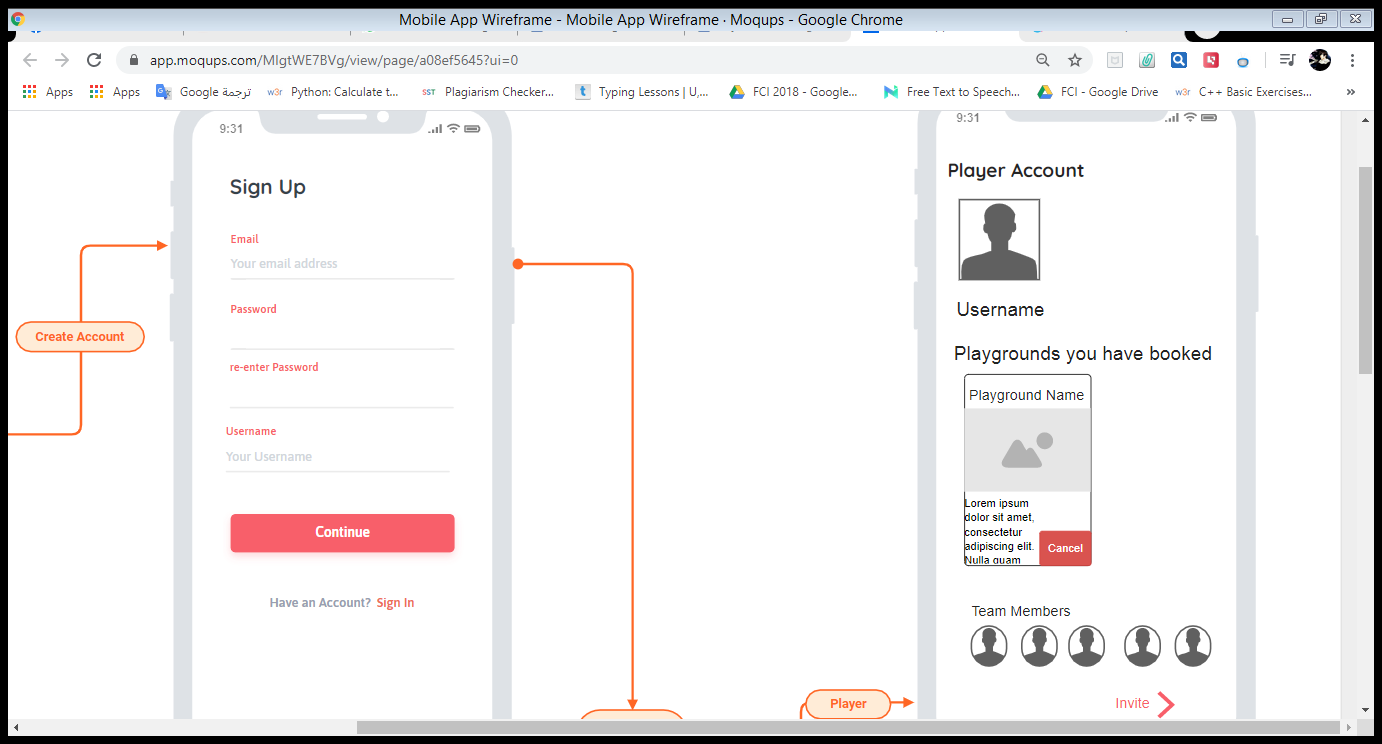
**ID : 01**

****

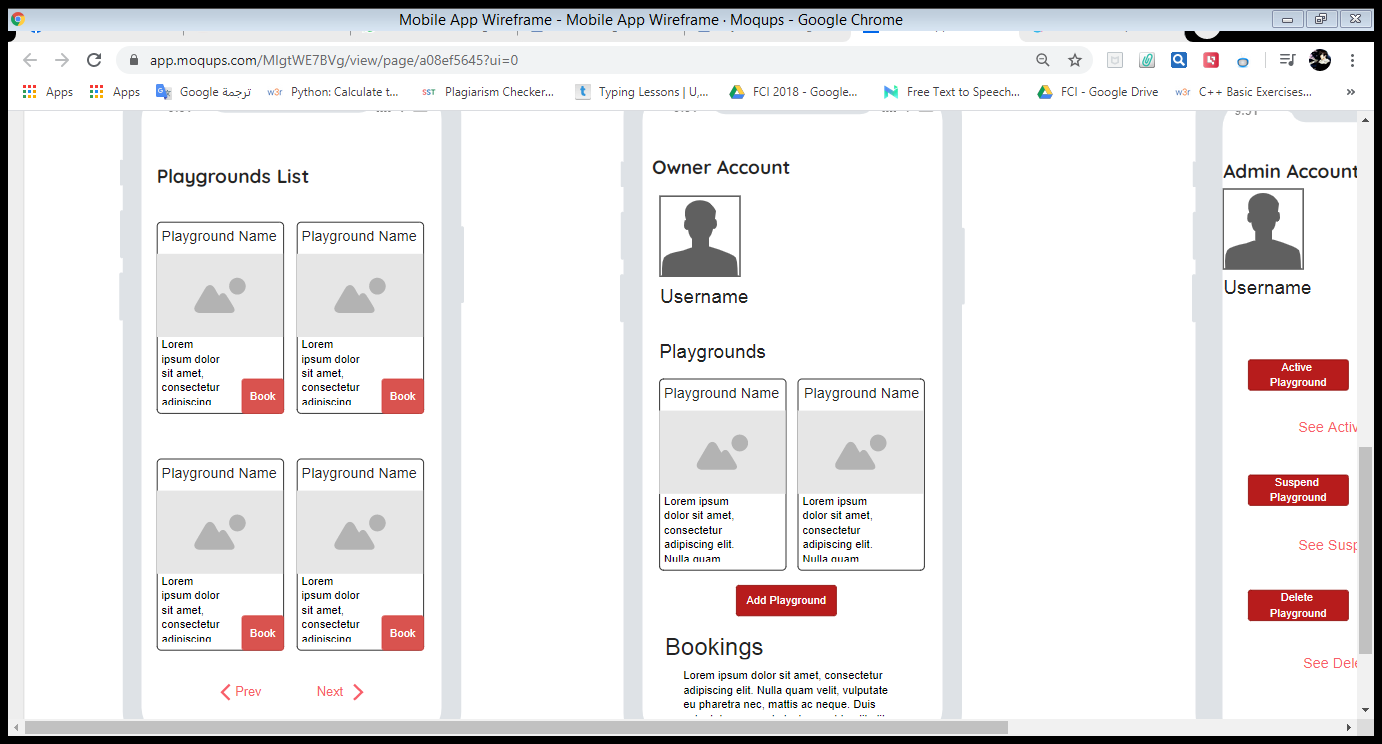
**ID: 02**

****

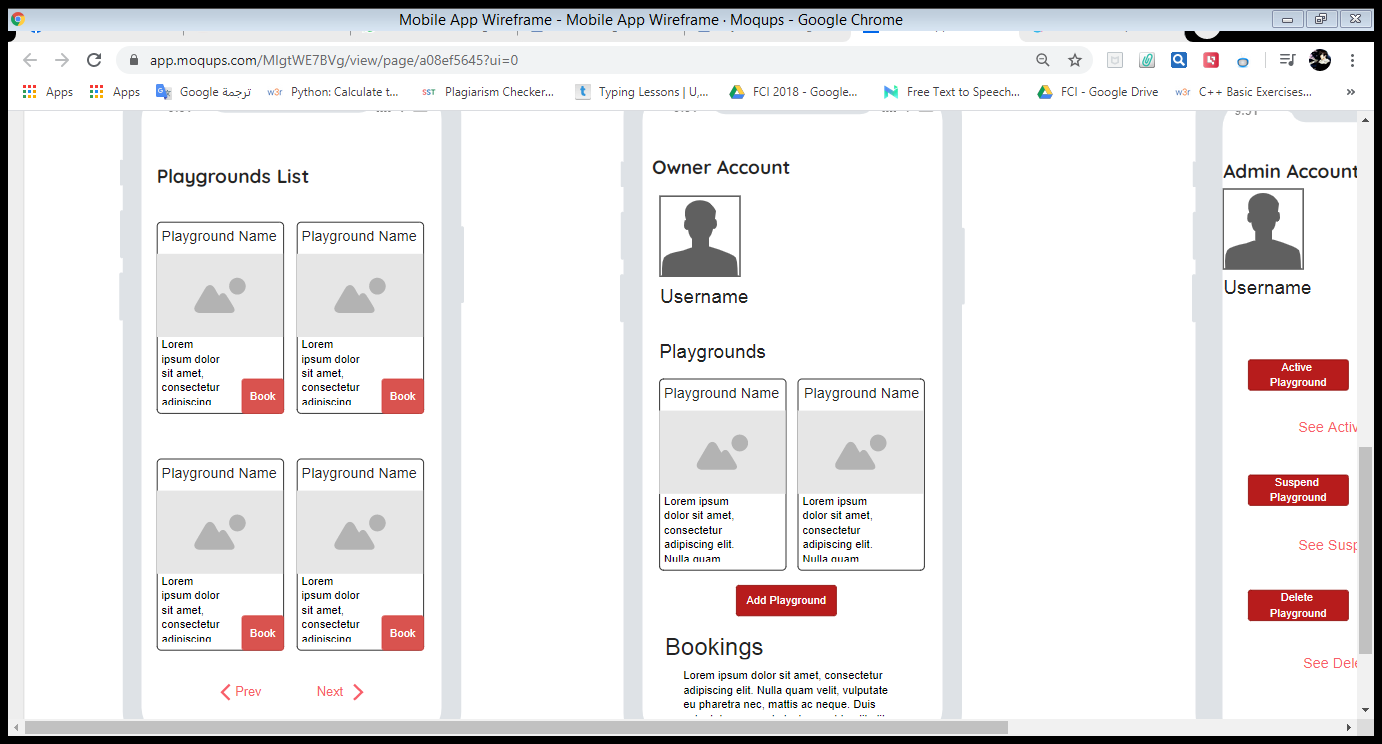
**ID: 03**

****

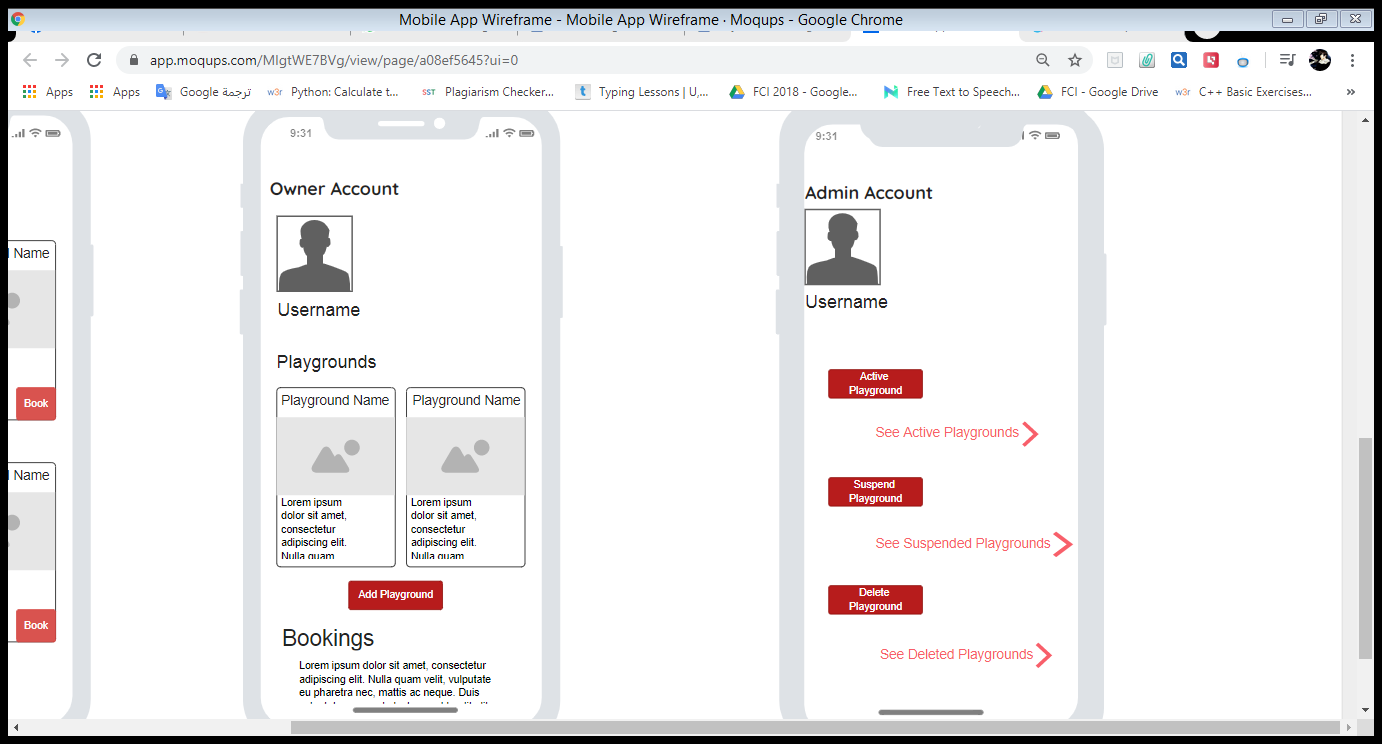
**ID 04:**

****

**ID 05:**

****

**ID 06:**

****

|  |  |  |
| --- | --- | --- |
| **Screen ID** | **Screen Name** | **Screen / Wireframe Description** |
| 01 | Sign in | This screen has a form to make the user sign in the GoFo App |
| 02 | Sign up | This screen has a form to make the user sign up the GoFo App |
| 03 | Player Profile | This screen shows the player profile that has the player info, bookings and team members. It allows to the player to cancel a booking and invite a team member. |
| 04 | Playgrounds List | This screen shows the playgrounds recorded into the system with its info. It also allows the players to book any playground. |
| 05 | Owner Profile | This screen shows the profile of the owner which includes the owner playgrounds. It also displays the owner playgrounds booking. |
| 06 | Admin Profile | This screen shows the admin profile info. It allows the admin to display the activated, suspended and non-activated playgrounds. It also allows the admin to activate, suspend or delete any playground. |

## Program implementation

<https://github.com/youssef2maher/GoFo.git>

Admin class:

**package** GoFoBackage;

**public** **class** Admin {

**private** String adminName = "Amir";

**private** String password = "321";

/\*\*

\* This function returns the admin name

\* **@return**

\*/

**public** String getAdminNAme() {

**return** adminName;

}

/\*\*

\* This function sets a value to the admin name

\* **@param** adminName

\*/

**public** **void** setAdminName(String adminName) {

**this**.adminName = adminName;

}

/\*\*

\* This function returns the admin password

\* **@return**

\*/

**public** String getPassword() {

**return** password;

}

/\*\*

\* This function sets a value to the admin password

\* **@param** password

\*/

**public** **void** setPassword(String password) {

**this**.password = password;

}

/\*\*

\* This is a default constructor

\*/

Admin()

{

}

/\*\*

\* This function login the admin into the system

\* **@param** name

\* **@param** pass

\* **@return**

\*/

**public** **boolean** login(String name, String pass) {

**if** (name.equals(adminName) && pass.equals(password)) {

**return** **true**;

}

**return** **false**;

}

/\*\*

\* This function deletes a playground

\* **@param** user

\* **@param** id

\*/

**public** **void** deleteGround(Register user, **int** id) {

**for**(**int** i = 0; i < user.owner.size(); i++) {

**for**(**int** j = 0; j < user.owner.get(i).ground.size(); j++) {

**if**(user.owner.get(i).ground.get(j).getGroundID() == id)

user.owner.get(i).ground.remove(j);

}

}

}

/\*\*

\* This function activates a playground

\* **@param** user

\* **@param** id

\*/

**public** **void** activateGround(Register user, **int** id) {

**for**(**int** i = 0; i < user.owner.size(); i++) {

**for**(**int** j = 0; j < user.owner.get(i).ground.size(); j++) {

**if**(user.owner.get(i).ground.get(j).getGroundID() == id)

user.owner.get(i).ground.get(j).setState("Active");

}

}

}

/\*\*

\* This function suspends a playground

\* **@param** user

\* **@param** id

\*/

**public** **void** suspendGround(Register user, **int** id) {

**for**(**int** i = 0; i < user.owner.size(); i++) {

**for**(**int** j = 0; j < user.owner.get(i).ground.size(); j++) {

**if**(user.owner.get(i).ground.get(j).getGroundID() == id)

user.owner.get(i).ground.get(j).setState("not");

}

}

}

/\*\*

\* This function approves a playground

\* **@param** user

\* **@param** id

\*/

**public** **void** approveGround(Register user, **int** id) {

**for**(**int** i = 0; i < user.owner.size(); i++) {

**for**(**int** j = 0; j < user.owner.get(i).ground.size(); j++) {

**if**(user.owner.get(i).ground.get(j).getGroundID() == id)

user.owner.get(i).ground.get(j).setState("Active");

}

}

}

/\*\*

\* This function displays the activated playgrounds

\* **@param** user

\*/

**public** **void** activatedGrounds(Register user) {

**for**(**int** i = 0; i < user.owner.size(); i++) {

**for**(**int** j = 0; j < user.owner.get(i).ground.size(); j++) {

**if**(user.owner.get(i).ground.get(j).getState() == "Active")

System.***out***.println(user.owner.get(i).ground.get(j));

}

}

}

/\*\*

\* This function displays the non-activated playgrounds

\* **@param** user

\*/

**public** **void** nonActivatedGrounds(Register user) {

**for**(**int** i = 0; i < user.owner.size(); i++) {

**for**(**int** j = 0; j < user.owner.get(i).ground.size(); j++) {

**if**(user.owner.get(i).ground.get(j).getState() == "notActive")

System.***out***.println(user.owner.get(i).ground.get(j));

}

}

}

/\*\*

\* This function displays the suspended playgrounds

\* **@param** user

\*/

**public** **void** suspendedGrounds(Register user) {

**for**(**int** i = 0; i < user.owner.size(); i++) {

**for**(**int** j = 0; j < user.owner.get(i).ground.size(); j++) {

**if**(user.owner.get(i).ground.get(j).getState() == "Suspend")

System.***out***.println(user.owner.get(i).ground.get(j));

}

}

}

}

Ewallet class:

**package** GoFoBackage;

**public** **class** Ewallet {

/\*\*

\* **@attributes**

\* balance integer

\*/

**private** **int** balance;

/\*\*

\* This function returns the balance value;

\* **@return**

\*/

**public** **int** getBalance() {

**return** balance;

}

/\*\*

\* This function sets a value to the balance

\* **@param** balance

\*/

**public** **void** setBalance(**int** balance) {

**this**.balance = balance;

}

/\*\*

\* This function allows the user to withdraw cash from his eWallet.

\* The function returns the withdrawing money if the user account

\* has enough money, otherwise, it returns -1.

\* **@param** valueWithdraw

\* **@return**

\*/

**public** **int** withdraw(**int** valueWithdraw) {

**if**(valueWithdraw <= balance){

balance -= valueWithdraw;

**return** valueWithdraw;

}

**else**{

**return** -1;

}

}

/\*\*

\* This function adds cash to the user eWallet.

\* **@param** valueDeposit

\*/

**public** **void** deposit (**int** valueDeposit) {

balance += valueDeposit;

}

/\*\*

\* This constructor sets an initial value to the user balance

\* in his eWallet.

\*/

**public** Ewallet() {

**this**.balance = 1000;

}

/\*\*

\* This parameterized constructor sets a value to the user

\* balance in his eWallet.

\* **@param** balance

\*/

**public** Ewallet(**int** balance) {

**this**.balance = balance;

}

}

Ground class:

**package** GoFoBackage;

**public** **class** Ground {

/\*\*

\* **@attributes**

\* groundName String

\* groundLocation String

\* groundArea float

\* groundID integer

\* state String

\* availableHour integer[][]

\* bookerID integer[]

\* price integer

\*/

**private** String groundName;

**private** String groundLocation;

**private** **float** groundArea;

**private** **int** groundID;

**private** String state = "notActive";

**private** **int** [][] availableHour = {{24,1},{1,2},{2,3},{3,4},{4,5},{5,6},{6,7},{7,8},{8,9},{9,10},{10,11},{11,12},{12,13},{13,14},{14,15},{15,16},{16,17},{17,18},{18,19},{19,20},{20,21},{21,22},{22,23},{23,24}};

**private** **int** [] bookerID = {-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1} ;

**private** **int** price;

/\*\*

\* This function returns the playground name.

\* **@return**

\*/

**public** String getGroundName() {

**return** groundName;

}

/\*\*

\* This function sets an initial value to the playground name.

\* **@param** groundName

\*/

**public** **void** setGroundName(String groundName) {

**this**.groundName = groundName;

}

/\*\*

\* This function returns the playground location.

\* **@return**

\*/

**public** String getGroundLocation() {

**return** groundLocation;

}

/\*\*

\* This function sets an initial value to the playground location.

\* **@param** groundLocation

\*/

**public** **void** setGroundLocation(String groundLocation) {

**this**.groundLocation = groundLocation;

}

/\*\*

\* This function returns the playground area.

\* **@return**

\*/

**public** **float** getGroundArea() {

**return** groundArea;

}

/\*\*

\* This function sets an initial value to the playground area.

\* **@param** groundArea

\*/

**public** **void** setGroundArea(**float** groundArea) {

**this**.groundArea = groundArea;

}

/\*\*

\* This function returns the playground ID.

\* **@return**

\*/

**public** **int** getGroundID() {

**return** groundID;

}

/\*\*

\* This function sets an initial value to the playground ID.

\* **@param** groundID

\*/

**public** **void** setGroundID(**int** groundID) {

**this**.groundID = groundID;

}

/\*\*

\* This function returns the playground state.

\* **@return**

\*/

**public** String getState() {

**return** state;

}

/\*\*

\* This function sets an initial value to the playground state.

\* **@param** state

\*/

**public** **void** setState(String state) {

**this**.state = state;

}

/\*\*

\* This function returns the playground available hours.

\* **@return**

\*/

**public** **int**[][] getAvailableHour() {

**return** availableHour;

}

/\*\*

\* This function sets an initial value to the playground available hours.

\* **@param** availableHour

\*/

**public** **void** setAvailableHour(**int**[][] availableHour) {

**this**.availableHour = availableHour;

}

/\*\*

\* This function sets an initial value to the playground available hours.

\* **@param** i

\* **@param** availableHour

\* **@param** availableHour2

\*/

**public** **void** setAvailableHour(**int** i,**int** availableHour,**int** availableHour2) {

**this**.availableHour[i][0] = availableHour;

**this**.availableHour[i][1] = availableHour2;

}

/\*\*

\* This function returns the playground booker ID.

\* **@return**

\*/

**public** **int** getBookerID(**int** i) {

**return** bookerID[i];

}

/\*\*

\* This function sets an initial value to the playground booker ID.

\* **@param** bookerID

\*/

**public** **void** setBookerID(**int** bookerID,**int** i) {

**this**.bookerID[i] = bookerID;

}

/\*\*

\* This function returns the playground price.

\* **@return**

\*/

**public** **int** getPrice() {

**return** price;

}

/\*\*

\* This function sets an initial value to the playground price.

\* **@param** price

\*/

**public** **void** setPrice(**int** price) {

**this**.price = price;

}

/\*\*

\* This function sets an initial value to the first and last hours of

\* the playground available hours for booking.

\* **@param** first

\* **@param** last

\*/

**public** **void** setHour(**int** first,**int** last) {

**for**(**int** i=0;i<24;i++)

{

availableHour[i][0] = 0;

availableHour[i][1] = 0;

}

**if**(first==24)

{

availableHour[0][0] = 24;

availableHour[0][1] = 1;

first=1;

}

**for**(**int** i=first;i<24;i++)

{

**if**(i==last)

**break**;

availableHour[i][0] = i;

availableHour[i][1] = i+1;

}

}

/\*\*

\* This constructor sets initial values to

\* the playground name, location, are and price.

\*/

**public** Ground() {

**this**.groundName = "";

**this**.groundLocation = "";

**this**.groundArea = 0;

**this**.price = 0;

}

/\*\*

\* This parameterized constructor sets values using its parameter to

\* the playground name, location, area, price and ID.

\* **@param** groundName

\* **@param** groundLocation

\* **@param** groundArea

\* **@param** price

\* **@param** ID

\*/

**public** Ground(String groundName, String groundLocation, **float** groundArea,**int** price,**int** ID) {

**this**.groundName = groundName;

**this**.groundLocation = groundLocation;

**this**.groundArea = groundArea;

**this**.price = price;

**this**.groundID = ID;

}

/\*\*

\* This function displays the available hours for booking for the playground.

\*/

**public** **void** displayAvailableHour() {

**for**(**int** i=0;i<24;i++)

{

**if**(availableHour[i][0] != 0 && bookerID[i] == -1 &&state.equalsIgnoreCase("active"))

System.***out***.print("{" + availableHour[i][0] + "," + availableHour[i][1] + "}");

}

System.***out***.println();

}

/\*\*

\* This function books the playground using the playground ID and

\* the first and last hours of the booking.

\* **@param** ID

\* **@param** first

\* **@param** last

\* **@return**

\*/

**public** **int** booking(**int** ID,**int** first,**int** last) {

**int** total = 0;

**if**(first==24){

bookerID[0] = ID;

first=1;

total+=price;

}

**for**(**int** i=first;i<24;i++){

**if**(i==last)

**break**;

bookerID[i] = ID;

total+=price;

}

**return** total;

}

/\*\*

\* This function displays the booking hours for the playground.

\*/

**public** **void** viewBooking() {

System.***out***.println(groundName+groundID);

System.***out***.print("{begin,last,playerID}");

**for**(**int** i=0;i<24;i++)

{

**if**(bookerID[i] != -1 || availableHour[i][0] == 25)

{

System.***out***.print("{" + availableHour[i][0] + "," + availableHour[i][1] + "," + bookerID[i] + "}");

System.***out***.println();

}

}

System.***out***.println("\n\n");

}

/\*\*

\*

\*/

@Override

**public** String toString() {

**return** "playground [groundName=" + groundName + ", groundLocation=" + groundLocation + ", groundArea="

+ groundArea + ", groundID=" + groundID + ", price=" + price + ", state=" + state + "]";

}

}

Ground owner Class:

package GoFoBackage;

import java.util.ArrayList;

import java.util.Scanner;

public class GroundOwner extends User {

public ArrayList<Ground> ground = new ArrayList<Ground>();

Scanner reader = new Scanner(System.in);

/\*\*

\* This function takes a playground name, location, area and price,

\* then it adds this playground in this the system,

\* and makes it belongs to the owner who added it.

\*/

public void addPlayground(String name, String location, float Area, int price) {

if(ground.size()==0)

{

Ground g= new Ground(name,location,Area,price,getID()+10);

ground.add(g);

}

else

{

Ground g= new Ground(name,location,Area,price,getID()+ground.get(ground.size()-1).getGroundID()+10);

ground.add(g);

}

}

/\*\*

\* This function allows the owner to make any update or change to their playgrounds,

\* changes could be for playground name, location, area and hours

\*/

public void updatePlayground(int groundNumber, String name, String location, float area, int first, int last,int change) {

if (change == 1) {

ground.get(groundNumber).setGroundName(name);

}

else if(change == 2){

ground.get(groundNumber).setGroundLocation(location);

}

else if(change == 3) {

ground.get(groundNumber).setGroundArea(area);

}

else if(change == 4){

ground.get(groundNumber).setHour(first, last);

}

}

/\*\*

\* This function display the playgrounds, belongs to this owner, info.

\* it also displays their available hours

\*/

public void viewPlaygrounds()

{

for(int i=0; i<ground.size(); i++)

{

System.out.println("[Ground [" + i + "]" + ground.get(i) + "]");

ground.get(i).displayAvailableHour();

}

}

/\*\*

\* This function displays the booking hours for their playgrounds

\*/

public void viewBooking()

{

for(int i=0;i<ground.size();i++)

{

ground.get(i).viewBooking();

}

}

/\*\*

\* This function displays the active playgrounds

\*/

public void viewActivePlaygrounds()

{

for(int i=0; i<ground.size(); i++)

{

if (ground.get(i).getState().equalsIgnoreCase("active"))

{

System.out.println("[ground=" + ground.get(i) + "]");

ground.get(i).displayAvailableHour();

}

}

}

/\*\*

\* this function to check if playground not booked

\* @param first

\* @param last

\* @param groundID

\* @return

\*/

public Boolean checkbooking(int first, int last, int groundID) {

Boolean flag=true;

if(first==24)

{

first=0;

}

for(int i=0; i<ground.size();i++)

{

if(ground.get(i).getGroundID() == groundID&&ground.get(i).getState().equalsIgnoreCase("active"))

{

for(int j=first; j<=last;j++)

{

if(ground.get(i).getBookerID(j)==-1)

{

flag=true;

}

else

{

flag=false;

break;

}

}

}

}

return flag;

}

/\*\*

\* This function allows the owner to book his own playgrounds and that's after

\* the player book it first.

\* The function displays all the playgrounds belongs to the owner and book

\* the playground by using its ID, the first and last hours of its available

\* hours, its cost and the owner ID.

\* The function check if the ID of the playground is right and then it books it

\* and pay its cost from the player eWallet to the owner one.

\* @param ID

\* @param first

\* @param last

\* @param cost

\* @param groundID

\* @return

\*/

public int booking(int ID, int first, int last, int cost, int groundID) {

for(int i=0; i<ground.size();i++)

{

if(ground.get(i).getGroundID() == groundID&&ground.get(i).getState().equalsIgnoreCase("active"))

{

groundID = i;

break;

}

}

ewallet.deposit(cost);

return ground.get(groundID).booking(ID, first, last);

}

/\*\*

\* This function displays the playgrounds of the owner in order of

\* its parameter location value.

\* @param location

\*/

public void Filtering(String location) {

for(int i=0;i<ground.size();i++)

{

if(ground.get(i).getGroundLocation().equalsIgnoreCase(location))

{

System.out.println(ground.get(i));

}

}

}

/\*\*

\* This function displays the playgrounds of the owner in order of

\* its parameter price value.

\* @param price

\*/

public void Filtering(int price) {

for(int i=0;i<ground.size();i++)

{

if(ground.get(i).getPrice() == price)

{

System.out.println(ground.get(i));

}

}

}

/\*\*

\*

\*/

public GroundOwner() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\*

\* @param iD

\* @param userName

\* @param mail

\* @param password

\* @param address

\* @param mobile

\* @param type

\*/

public GroundOwner(int iD, String userName, String mail, String password, String address, String mobile, String type) {

super(iD, userName, mail, password, address, mobile, type);

// TODO Auto-generated constructor stub

}

/\*\*

\*

\*/

@Override

public String toString() {

return "owner" + "[ID=" + getID() + ", userName=" + getUserName() + ", mail=" + getMail() + ", password=" + getPassword() + ", address="

+ getAddress() + ", mobile=" + getMobile() + ", type=" + getType()+ "]"+"[ground=" + ground + "]";

}

}

Player class:

**package** GoFoBackage;

**public** **class** Player **extends** User {

/\*\*

\* **@attributes**

\* teamName String

\* teamPlayers User[]

\* playerNumberInTeam integer

\*/

**private** String teamName = "Team";

**public** User teamPlayers[] = **new** Player[11];

**private** **int** playersNumber = 0;

/\*\*

\* This function returns the team name.

\* **@return**

\*/

**public** String getTeamName(){

**return** teamName;

}

/\*\*

\* This function sets the team Name.

\* **@param** teamName

\*/

**public** **void** setTeamName(String teamName) {

**this**.teamName = teamName;

}

/\*\*

\* This function returns the players number in the team.

\* **@return**

\*/

**public** **int** getPlayersNumber() {

**return** playersNumber;

}

/\*\*

\* This function returns the players number in the team.

\* **@param** playersNumber

\*/

**public** **void** setPlayersNumber(**int** playersNumber) {

**this**.playersNumber = playersNumber;

}

/\*\*

\* This function displays all the approved playgrounds to the player.

\* **@param** users

\*/

**public** **void** viewPlayground(Register users) {

**for**(**int** i=0;i< users.owner.size(); i++)

{

System.***out***.println("Owner ID [" + users.owner.get(i).getID() + "]");

users.owner.get(i).viewActivePlaygrounds();

}

}

/\*\*

\* This function filters the playground menu to display all the playground

\* in order of the player location, or the needed price.

\* **@param** users

\* **@param** filterBy

\* **@param** price

\*/

**public** **void** filter(Register users, String filterBy, **int** price) {

**if** (filterBy.equalsIgnoreCase("location")) {

**for**(**int** i=0;i< users.owner.size(); i++){

users.owner.get(i).Filtering(getAddress());

}

}

**else** **if**(filterBy.equalsIgnoreCase("price")){

**for**(**int** i=0;i< users.owner.size(); i++){

users.owner.get(i).Filtering(price);

}

}

}

/\*\*

\* This function allows the player to a playground using the owner ID,

\* then the player determine the first and last hours for his booking.

\* The function checks if the owner ID is right, then it checks the player

\* eWallet and see if he is able to pay its cost.

\* **@param** users

\* **@param** ownerID

\* **@param** first

\* **@param** last

\*/

**public** **int** bookPlayground(Register users, **int** ownerID, **int** first, **int** last, **int** groundID) {

**int** index = 0;

**for**(**int** i = 0; i < users.owner.size(); i++)

{

**if**(users.owner.get(i).getID()== ownerID){

index = i;

**break**;

}

index = i + 1;

}

**if**(index == users.owner.size()){

**return** -2;

}

**else** {

**if**(users.owner.get(index).checkbooking( first, last, groundID)==**true**)

{

**int** cost;

**if**(first == 24){

cost = last;

}

**else** {

cost = last - first;

}

**if**(ewallet.withdraw(cost) != -1) {

**return** users.owner.get(index).booking(getID(),first,last,cost, groundID);

}

**else** {

**return** -1;

}

}

**else**

{

System.***out***.println("your choose booked hour");

**return** -1;

}

}

}

/\*\*

\* this function to print booking ground at player

\* **@param** user

\*/

**public** **void** printbooking(Register user)

{

**for**(**int** i=0;i<user.owner.size();i++)

{

**for**(**int** j=0;j<user.owner.get(i).ground.size();j++)

{

**for**(**int** k=0;k<24;k++)

{

**if**(user.owner.get(i).ground.get(j).getBookerID(k) == getID()) {

System.***out***.println(user.owner.get(i).ground.get(j));

**break**;

}

}

}

}

}

/\*\*

\* This function cancel a playground booking

\* **@param** user

\* **@param** ID

\*/

**public** **void** cancelPlayground(Register user, **int** ID) {

**for**(**int** i=0;i<user.owner.size();i++)

{

**for**(**int** j=0;j<user.owner.get(i).ground.size();j++)

{

**if**(user.owner.get(i).ground.get(j).getGroundID()==ID)

{ **for**(**int** k=0;k<24;k++)

{

**if**(user.owner.get(i).ground.get(j).getBookerID(k)==getID())

{

user.owner.get(i).ground.get(j).setBookerID(-1,k);

**if**(k==0)

{

user.owner.get(i).ground.get(j).setAvailableHour(k, 24, 1);

}

**else**

{

user.owner.get(i).ground.get(j).setAvailableHour(k, k+1, k+2);

}

ewallet.deposit(user.owner.get(i).ground.get(j).getPrice());

user.owner.get(i).ewallet.withdraw(user.owner.get(i).ground.get(j).getPrice());

}

}

}

}

}

}

/\*\*

\* This function allows the player to add other players in his team using

\* the player user name.

\* **@param** playerName

\* **@param** users

\*/

**public** **boolean** addPlayer(String playerName, Register users) {

**for** (**int** i = 0; i < users.ID; i++) {

**if** (users.player.get(i).getUserName().equalsIgnoreCase(playerName)) {

teamPlayers[playersNumber] = users.player.get(i);

playersNumber++;

**return** **true**;

}

}

**return** **false**;

}

/\*\*

\*

\* **@param** playerName

\*/

**public** **void** sendInvitation(String playerName) {

}

/\*\*

\*

\*/

**public** Player() {

**super**();

// **TODO** Auto-generated constructor stub

}

/\*\*

\*

\* **@param** iD

\* **@param** userName

\* **@param** mail

\* **@param** password

\* **@param** address

\* **@param** mobile

\* **@param** type

\*/

**public** Player(**int** iD, String userName, String mail, String password, String address, String mobile, String type) {

**super**(iD, userName, mail, password, address, mobile, type);

// **TODO** Auto-generated constructor stub

}

}

Register Class:

**package** GoFoBackage;

**import** java.util.ArrayList;

**public** **class** Register {

/\*\*

\* **@attributes**

\* owner ArrayList<GroundOwner>

\* player ArrayList<Player>

\* ID integer

\* admin Admin

\*/

**public** ArrayList<GroundOwner> owner = **new** ArrayList<GroundOwner>();

**public** ArrayList<Player> player = **new** ArrayList<Player>();

**public** **int** ID = 0;

**public** Admin admin = **new** Admin();

/\*\*

\* This function allows the user to sign up in the program by creating his profile.

\* The function creating a new profile for the user and store his info. which is

\* (user name, mail, password, address, mobile number, type).

\* It checks if the user name entered is unique, and if it is, it returns true,

\* otherwise it returns false.

\* **@param** name

\* **@param** mail

\* **@param** password

\* **@param** address

\* **@param** mobile

\* **@param** type

\* **@return**

\*/

**public** **boolean** signUp(String name, String mail, String password, String address, String mobile, String type){

**try** {

**for** (**int** i = 0; i < owner.size(); i++) {

**if** (owner.get(i).getUserName().equalsIgnoreCase(name)) {

**return** **false**;

}

}

**for** (**int** i = 0; i < player.size(); i++) {

**if** (player.get(i).getUserName().equalsIgnoreCase(name)) {

**return** **false**;

}

}

} **catch** (Exception e) {

// **TODO**: handle exception

}

**if**(type.equalsIgnoreCase("Owner")){

GroundOwner newUser = **new** GroundOwner(ID,name,mail,password,address,mobile,type);

owner.add(newUser);

ID++;

}

**else** **if**(type.equalsIgnoreCase("Player")){

Player newUser = **new** Player(ID,name,mail,password,address,mobile,type);

player.add(newUser);

ID++;

}

**return** **true**;

}

/\*\*

\* This function allows the user to login with his user name and password of

\* his account that already have been created.

\* The function returns the ID number of the user account, otherwise,

\* it returns -1.

\* **@param** mail

\* **@param** password

\* **@return**

\*/

**public** **int** login(String mail, String password, String type) {

**try** {

**if** (type.equalsIgnoreCase("owner")) {

**for** (**int** i = 0; i < owner.size(); i++) {

**if** (owner.get(i).getMail().equalsIgnoreCase(mail)&&owner.get(i).getPassword().equalsIgnoreCase(password)) {

**return** i;

}

}

}

**else** {

**for** (**int** i = 0; i < player.size(); i++) {

**if** (player.get(i).getMail().equalsIgnoreCase(mail)&&player.get(i).getPassword().equalsIgnoreCase(password)) {

**return** i;

}

}

}

} **catch** (Exception e) {

// **TODO**: handle exception

}

**return** -1;

}

}

User class:

**package** GoFoBackage;

**public** **class** User {

/\*\*

\* **@attributes**

\* ID integer

\* userName String

\* mail String

\* password String

\* address String

\* mobile String

\* type String

\*/

**private** **int** ID;

**private** String userName;

**private** String mail;

**private** String password;

**private** String address;

**private** String mobile;

**private** String type;

**public** Ewallet ewallet = **new** Ewallet();

/\*\*

\* This function returns the ID of the user.

\* **@return**

\*/

**public** **int** getID() {

**return** ID;

}

/\*\*

\* This function sets a value of the ID of the user.

\* **@param** iD

\*/

**public** **void** setID(**int** iD) {

ID = iD;

}

/\*\*

\* This function returns the user name of the user.

\* **@return**

\*/

**public** String getUserName() {

**return** userName;

}

/\*\*

\* This function sets a value of the user name of the user.

\* **@param** userName

\*/

**public** **void** setUserName(String userName) {

**this**.userName = userName;

}

/\*\*

\* This function returns the mail of the user.

\* **@return**

\*/

**public** String getMail() {

**return** mail;

}

/\*\*

\* This function sets a value of the mail of the user.

\* **@param** mail

\*/

**public** **void** setMail(String mail) {

**this**.mail = mail;

}

/\*\*

\* This function returns the password of the user.

\* **@return**

\*/

**public** String getPassword() {

**return** password;

}

/\*\*

\* This function sets a value of the password of the user.

\* **@param** password

\*/

**public** **void** setPassword(String password) {

**this**.password = password;

}

/\*\*

\* This function returns the address of the user.

\* **@return**

\*/

**public** String getAddress() {

**return** address;

}

/\*\*

\* This function sets a value of the address of the user.

\* **@param** address

\*/

**public** **void** setAddress(String address) {

**this**.address = address;

}

/\*\*

\* This function returns the mobile number of the user.

\* **@return**

\*/

**public** String getMobile() {

**return** mobile;

}

/\*\*

\* This function sets a value of the mobile number of the user.

\* **@param** mobile

\*/

**public** **void** setMobile(String mobile) {

**this**.mobile = mobile;

}

/\*\*

\* This function returns the type of the user.

\* **@return**

\*/

**public** String getType() {

**return** type;

}

/\*\*

\* This function sets a value of the type of the user.

\* **@param** type

\*/

**public** **void** setType(String type) {

**this**.type = type;

}

/\*\*

\* This is a default constructor

\*/

**public** User()

{

}

/\*\*

\* This parameterized constructor sets initial values to the info. of the user which is

\* (ID, user name, mail, password, address, mobile number, type).

\* **@param** iD

\* **@param** userName

\* **@param** mail

\* **@param** password

\* **@param** address

\* **@param** mobile

\* **@param** type

\*/

**public** User(**int** iD, String userName, String mail, String password, String address, String mobile, String type) {

ID = iD;

**this**.userName = userName;

**this**.mail = mail;

**this**.password = password;

**this**.address = address;

**this**.mobile = mobile;

**this**.type = type;

}

/\*\*

\*

\*/

@Override

**public** String toString() {

**return** "User [ID=" + ID + ", userName=" + userName + ", mail=" + mail + ", password=" + password + ", address="

+ address + ", mobile=" + mobile + ", type=" + type + "]";

}

}

Main Class:

**package** GoFoBackage;

**import** java.util.Scanner;

**public** **class** UI {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

//This is a welcome phrase

System.***out***.println("\t\t\t Welcome to GoFo!!");

//This is a scanner to get input through the console

Scanner scan = **new** Scanner(System.***in***);

/\*\*

\* Register Section

\*/

Register regist = **new** Register(); //This an object from Register class

//These are two initial accounts

regist.signUp("Diab", "Diab@gmail.com", "235711", "Hdayk El-Maady", "0123456789", "owner");

regist.owner.get(0).addPlayground("A", "Maady", 100, 10);

regist.owner.get(0).addPlayground("B", "Salam", 200, 20);

regist.owner.get(0).addPlayground("C", "Giza", 300, 30);

regist.signUp("Youssef", "Youssef@gmail.com", "131719", "Dar El-Salam", "01113453457", "player");

regist.signUp("Ahmed", "Ahmed@gmail.com", "131719", "Dar El-Salam", "01113453457", "player");

regist.signUp("Mohamed", "Mohamed@gmail.com", "131719", "Dar El-Salam", "01113453457", "player");

regist.signUp("Ali", "Ali@gmail.com", "131719", "Dar El-Salam", "01113453457", "player");

////////////////////////////////

**while** (**true**) {

//To choose between login or sign up

System.***out***.print("\n\n 1. Sign up \n 2. Login \n 3. Admin \n -> ");

**int** var1 = scan.nextInt();

**boolean** registerState; //This variable to store the register state if true or false

/\*

\* Sign-up Phase

\*/

**if** (var1 == 1) {

System.***out***.println("\n\n -- Register a New Account -- \n");

System.***out***.print("\n\t Enter User Name: ");

String userNameVar = scan.next();

System.***out***.print("\n\t Enter Mail: ");

String mailVar = scan.next();

System.***out***.print("\n\t Enter Password: ");

String passwordVar = scan.next();

System.***out***.print("\n\t Enter Address: ");

String addressVar = scan.next();

System.***out***.print("\n\t Enter Mobile Number: ");

String mobileVar = scan.next();

System.***out***.print("\n\t Enter Type (Player or Owner): ");

String typeVar = scan.next();

registerState = regist.signUp(userNameVar, mailVar, passwordVar, addressVar, mobileVar, typeVar);

//To see check if the account has been signed up or not

**if** (registerState) {

System.***out***.println("\n\n\t\t \*\* The account has been signed up \*\* ");

}**else** {

**while**(!registerState) {

System.***out***.println("\n\n\t\t \*\* The User Name is already existed \*\* ");

System.***out***.print("\n\t Enter User Name: ");

userNameVar = scan.next();

registerState = regist.signUp(userNameVar, mailVar, passwordVar, addressVar, mobileVar, typeVar);

}

System.***out***.println("\n\n\t\t \*\* The account has been signed up \*\* ");

}

}

/\*

\* Login Phase

\*/

String typeVar = " "; **int** loginAccID = 0;

**if** (var1 != 3) {

System.***out***.println("\n\n -- Login with an Existed Account -- \n");

//To check if the user is owner or player

System.***out***.print("\n Enter Account Type (Owner or Player): ");

typeVar = scan.next();

System.***out***.print("\n\t Enter User mail: ");

String userNameVar = scan.next();

System.***out***.print("\n\t Enter Password: ");

String passwordVar = scan.next();

loginAccID = regist.login(userNameVar, passwordVar, typeVar);

**while**(loginAccID == -1) {

System.***out***.println("\n\n\t\t \*\* Your User Name or Password isn't Correct \*\* \n");

System.***out***.print("\n\t Enter User mail: ");

userNameVar = scan.next();

System.***out***.print("\n\t Enter Password: ");

passwordVar = scan.next();

loginAccID = regist.login(userNameVar, passwordVar, typeVar);

}

System.***out***.println("\n\n\t\t \*\* You'er In \*\* ");

}

/\*\*

\* Playground Owner Main Menu Section

\*/

**if** (typeVar.equalsIgnoreCase("owner")) {

GroundOwner ownerObj = regist.owner.get(loginAccID);//The user account object

**while** (**true**) {

//Displaying the menu for the owner and make him choose an option

System.***out***.println("\n\n\t\t\t Welcome " + ownerObj.getUserName() + "\n");

System.***out***.print(" [1] Add Playground \n"

+ " [2] Update Playground \n"

+ " [3] See Booking Requests \n"

+ " [4] eWallet \n"

+ " [5] Log out \n ->");

**int** option = scan.nextInt();

/\*

\* Add Playground Option

\*/

**if** (option == 1) {

System.***out***.println("\n\n -- Add Playground -- \n");

System.***out***.print("\t Enter Playground Name: ");

String playgroundNameVar = scan.next();

System.***out***.print("\t Enter Playground Location: ");

String playgroundLocationVar = scan.next();

System.***out***.print("\t Enter Playground Area: ");

**float** playgroundAreaVar = scan.nextFloat();

System.***out***.print("\t Enter Playground Price: ");

**int** playgroundPriceVar = scan.nextInt();

ownerObj.addPlayground(playgroundNameVar, playgroundLocationVar, playgroundAreaVar, playgroundPriceVar);

}

/\*

\* Update Playground Option

\*/

**else** **if** (option == 2) {

System.***out***.println("\n\n -- Update Playground -- \n");

//Displaying all the owner playgrounds

ownerObj.viewPlaygrounds();

System.***out***.print("\n\n\t Enter Playground number: \n");

**int** playgroundIDVar = scan.nextInt();

**while**(**true**) {

System.***out***.print(" (1) Change Playground Name \n"

+ " (2) Change Location \n"

+ " (3) Change Area \n"

+ " (4) Change Booking Hours \n"

+ " (5) Go Back \n -> ");

**int** change = scan.nextInt();

String playgroundNameVar = ownerObj.ground.get(playgroundIDVar).getGroundName();

String playgroundLocationVar = ownerObj.ground.get(playgroundIDVar).getGroundLocation();

**float** playgroundAreaVar = ownerObj.ground.get(playgroundIDVar).getGroundArea();

**int** startHourVar = 0, endHourVar = 24;

**if** (change == 1) {

System.***out***.print(" Enter the New Playground Name: ");

playgroundNameVar = scan.next();

ownerObj.updatePlayground(playgroundIDVar, playgroundNameVar, playgroundLocationVar, playgroundAreaVar, startHourVar, endHourVar,change);

}

**else** **if** (change == 2) {

System.***out***.print(" Enter the New Playground Location: ");

playgroundLocationVar = scan.next();

ownerObj.updatePlayground(playgroundIDVar, playgroundNameVar, playgroundLocationVar, playgroundAreaVar, startHourVar, endHourVar,change);

}

**else** **if** (change == 3) {

System.***out***.print(" Enter the New Playground Area: ");

playgroundAreaVar = scan.nextFloat();

ownerObj.updatePlayground(playgroundIDVar, playgroundNameVar, playgroundLocationVar, playgroundAreaVar, startHourVar, endHourVar,change);

}

**else** **if** (change == 4) {

System.***out***.print(" Enter the New Playground Area: ");

startHourVar = scan.nextInt();

endHourVar = scan.nextInt();

ownerObj.updatePlayground(playgroundIDVar, playgroundNameVar, playgroundLocationVar, playgroundAreaVar, startHourVar, endHourVar,change);

}

**else** **if** (change == 5) {

**break**;

}

}

}

/\*

\* Booking Requests Option

\*/

**else** **if** (option == 3) {

System.***out***.println("\n\n -- Booking Requests -- \n");

ownerObj.viewBooking();

}

/\*

\* eWallet Option

\*/

**else** **if** (option == 4) {

System.***out***.println("\n\n -- eWallet -- \n");

**while** (**true**) {

System.***out***.print("\n\n (1) Deposite \n"

+ " (2) Withdraw \n"

+ " (3) Balance \n"

+ " (4) Done! \n -> ");

**int** operation = scan.nextInt();

**if** (operation == 1) {

System.***out***.print(" Enter the Deposited Value: ");

**int** dopsiteValueVar = scan.nextInt();

ownerObj.ewallet.deposit(dopsiteValueVar);

}

**else** **if** (operation == 2) {

System.***out***.print(" Enter the Withdrawn Value: ");

**int** withdrawValueVar = scan.nextInt();

ownerObj.ewallet.withdraw(withdrawValueVar);

}

**else** **if** (operation == 3) {

System.***out***.print(" Your Balance: " + ownerObj.ewallet.getBalance());

}

**else** **if** (operation == 4) {

**break**;

}

}

}

/\*

\* Log out Option

\*/

**else** **if** (option == 5) {

**break**;

}

}

}

/\*\*

\* Player Main Menu Section

\*/

**else** **if** (typeVar.equalsIgnoreCase("player")) {

Player playerObj = regist.player.get(loginAccID);// The Player Account Object

**while** (**true**) {

//Displaying the Main Menu to the Player and choose operation to do

System.***out***.println("\n\n\t\t\t Welcome " + playerObj.getUserName());

System.***out***.print(" [1] View Playgrounds \n "

+ " [2] Book a Playground \n "

+ " [3] Cancel a Booking \n "

+ " [4] Add Player to your Team \n "

+ " [5] View Team \n "

+ " [6] Log out \n -> ");

**int** option = scan.nextInt();

/\*

\* View Playgrounds Option

\*/

**if** (option == 1) {

System.***out***.println("\n\n -- View Playgrounds -- \n");

playerObj.viewPlayground(regist);

}

/\*

\* Book a Playground Option

\*/

**else** **if** (option == 2) {

System.***out***.println("\n\n -- Book a Playground -- \n");

playerObj.viewPlayground(regist);

System.***out***.print(" Enter Owner ID: ");

**int** ownerIDVar = scan.nextInt();

System.***out***.print(" Enter Playground Number: ");

**int** playgourndIDVar = scan.nextInt();

System.***out***.print(" Enter the Starting Hour: ");

**int** firstVar = scan.nextInt();

System.***out***.print(" Enter the Ending Hour: ");

**int** lastVar = scan.nextInt();

**int** bookingState = playerObj.bookPlayground(regist, ownerIDVar, firstVar, lastVar, playgourndIDVar);

**if** (bookingState == -2) {

System.***out***.println("\n\n \*\* The Owner ID not Found \*\* ");

}

**else** **if** (bookingState == -1) {

System.***out***.println("\n\n \*\* Your Current Balance does't allow the booking \*\* ");

}

**else** {

System.***out***.println("\n\n \*\* Done!! \*\* ");

}

}

/\*

\* Cancel A Booking Option

\*/

**else** **if** (option == 3) {

System.***out***.println("\n\n -- Cancel a Booking -- \n");

playerObj.printbooking(regist);

**int** ID;

System.***out***.println("Enter ID of Ground You Want");

ID = scan.nextInt();

playerObj.cancelPlayground(regist, ID);

}

/\*

\* Add a Player to your Team Option

\*/

**else** **if** (option == 4) {

System.***out***.println("\n\n -- Add a Player to you Team -- \n");

System.***out***.println(" Enter the Team Mate Name: ");

String teamMateNameVar = scan.next();

**boolean** addPlayerState = playerObj.addPlayer(teamMateNameVar, regist);

**if** (addPlayerState) {

System.***out***.println("\n\n \*\* Done!! \*\* ");

}

**else** {

System.***out***.println("\n\n \*\* The Player not Found \*\* ");

}

}

/\*

\* View Team Option

\*/

**else** **if** (option == 5) {

System.***out***.println("\n\n -- View Team -- \n");

**int** playerNumberInTeam = 0;

System.***out***.println(" Team Members: ");

**for** (**int** i = 0; i < playerObj.getPlayersNumber(); i++) {

System.***out***.println(" [" + playerNumberInTeam++ + "] " + playerObj.teamPlayers[i]);

}

}

/\*

\* Log out Option

\*/

**else** **if** (option == 6) {

**break**;

}

}

}

/\*\*

\* Admin Main Menu

\*/

**else** {

Admin adminObj = regist.admin;

System.***out***.print("\n\n Enter Admin Name: ");

String adminNameVar = scan.next();

System.***out***.print(" Enter Password: ");

String adminPasswordVar = scan.next();

**boolean** loginState = adminObj.login(adminNameVar, adminPasswordVar);

**if** (!loginState) {

System.***out***.println("\n\n \*\* Wrong Name or Password \*\* ");

}

**while** (loginState) {

System.***out***.println("Welcome Admin");

System.***out***.print(" [1] View Activated Playgrounds \n "

+ " [2] View Non-activated Playgrounds \n "

+ " [3] View Suspended Playgrounds \n "

+ " [4] Delete Playground \n "

+ " [5] Activate Playground \n "

+ " [6] Suspend Playground \n "

+ " [7] Log Out \n -> ");

**int** option = scan.nextInt();

/\*

\* View Activated Playgrounds Option

\*/

**if** (option == 1) {

System.***out***.println("\n\n -- View Activated Playgrounds -- \n");

adminObj.activatedGrounds(regist);

}

/\*

\* View Non-activated Playgrounds Option

\*/

**else** **if** (option == 2) {

System.***out***.println("\n\n -- View Non-activated Playgrounds -- \n");

adminObj.nonActivatedGrounds(regist);

}

/\*

\* View Suspended Playgrounds Option

\*/

**else** **if** (option == 3) {

System.***out***.println("\n\n -- View Suspended Playgrounds -- \n");

adminObj.suspendedGrounds(regist);

}

/\*

\* Delete Playgrounds Option

\*/

**else** **if** (option == 4) {

System.***out***.println("\n\n -- Delete Playground -- \n");

System.***out***.print(" Enter the Playground ID: ");

**int** deletedPlaygroundIDVar = scan.nextInt();

adminObj.deleteGround(regist, deletedPlaygroundIDVar);

System.***out***.println("\n\n \*\* Done \*\* ");

}

/\*

\* Activate Playgrounds Option

\*/

**else** **if** (option == 5) {

System.***out***.println("\n\n -- Activate Playground -- \n");

System.***out***.print(" Enter the Playground ID: ");

**int** activatedPlaygroundIDVar = scan.nextInt();

adminObj.activateGround(regist, activatedPlaygroundIDVar);

System.***out***.println("\n\n \*\* Done \*\* ");

}

/\*

\* Suspend Playgrounds Option

\*/

**else** **if** (option == 6) {

System.***out***.println("\n\n -- Suspend Playground -- \n");

System.***out***.print(" Enter the Playground ID: ");

**int** suspendedPlaygroundIDVar = scan.nextInt();

adminObj.suspendGround(regist, suspendedPlaygroundIDVar);

System.***out***.println("\n\n \*\* Done \*\* ");

}

/\*

\* Log Out Option

\*/

**else** **if** (option == 7) {

**break**;

}

}

}

}

}

}

# Tools

* Eclipse java 2019
* Visual-Paradigm
* [www.moqups.com](http://www.moqups.com)
* Lucidchart.com

# Links

GitHub Repo:

<https://github.com/youssef2maher/GoFo.git>

Code Implementation

<https://drive.google.com/drive/folders/1DeAgEhmExb3mjY4ZJbhr9sjZkIv_sUmu?usp=sharing>

Code Implementation Video

<https://drive.google.com/drive/folders/1IzgcVgVewLordejQ3GdgKKtO563UugbR?usp=sharing>

# Ownership Report

|  |  |
| --- | --- |
| **Item** | **Owners** |
| Class Diagram | Mohamed Gamal Diab  Youssef Maher Maher |
| Sequence Diagram | Alamir Hassan Younis  Ali Mohamed Fathy |
| UI | Youssef Maher Maher  Ali Mohamed Fathy |
| Implementation | Youssef Maher Maher  Mohamed Gamal Diab  Ali Mohamed Fathy  Alamir Hassan Younis |

# 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>
* Example wireframes and designs (you can contact the author for questions) <http://malakumar.com/wp-content/uploads/2018/12/MalaKumar_SampleWireframes-1.pdf>

# Authors

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