Cairo University

Faculty of Computers and Information

CS251

Software Engineering I

GoFo

Software Design Specifications

And Implementation

Version 2.0

Team Names and Emails

June & 2020

CS251: Phase 2 – <Tankers>
Project: <GoFo>

Software Design Specification

Contents

CS251: Phase 2 – <Tankers>
Project: <GoFo>

Software Design Specification

- **Team**

ID	Name	Email	Mobile
20180128	Seif Mosaad Abd El-Fattah	Eng.seifmosaad735@gmail.com	01200372782
20180413	Ahmed Nabil Mohamed Salah	anabilsalah@gmail.com	01004158778
20180083	Habiba Amr Mohamed	Habibaamr350@gmail.com	01120600350
20180208	Marina Moheb Nafee		01288230559

CS251: Phase 2 – <Tankers>
Project: <GoFo>

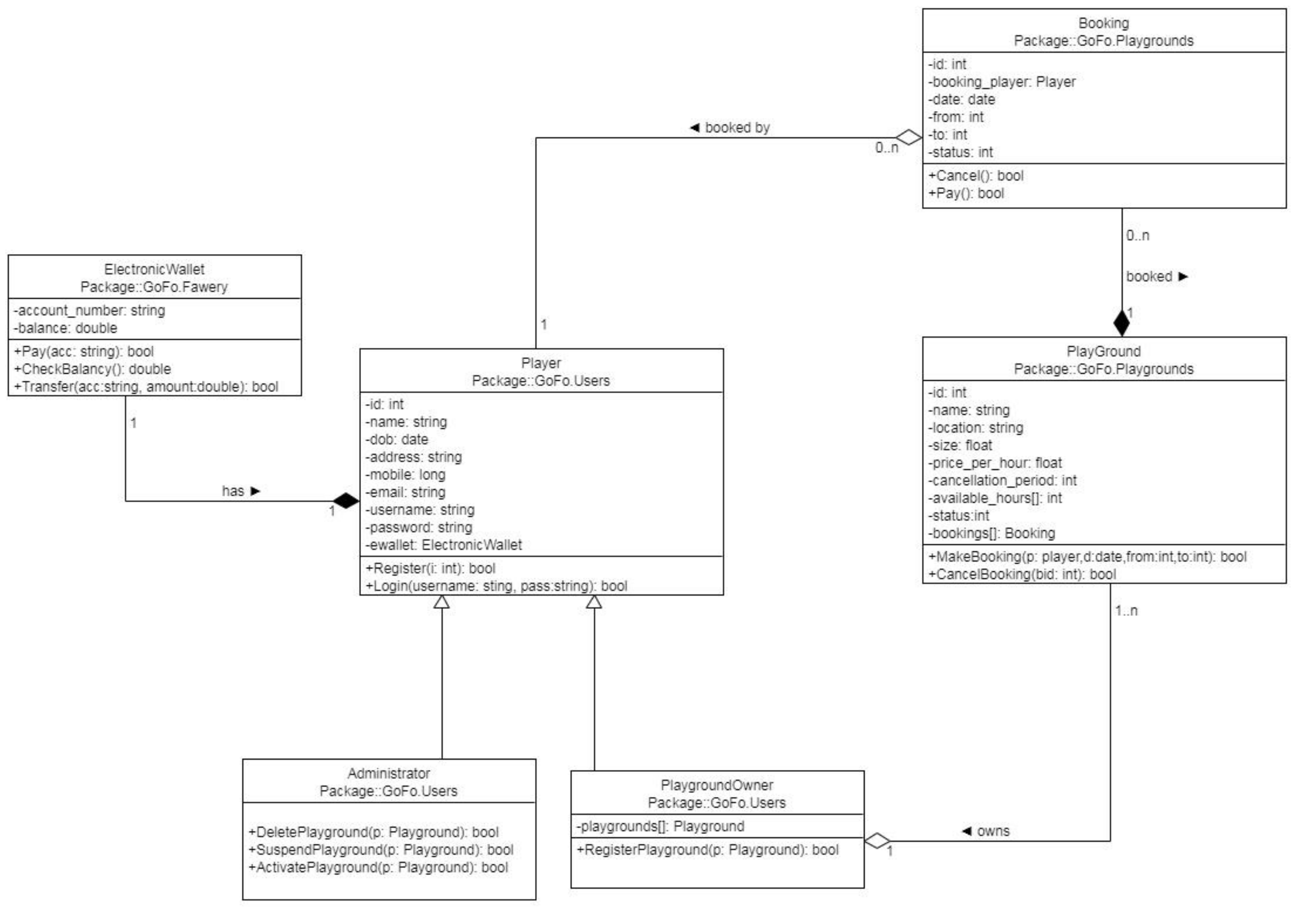
Software Design Specification

Document Purpose and Audience

- **This is an SDS for developers and software engineers.**
- **This document explains how GOFO system operations work.**
- **Mangers, Software Engineers, Developers.**

System Models

I. Class Diagram(s)



Software Design Specification

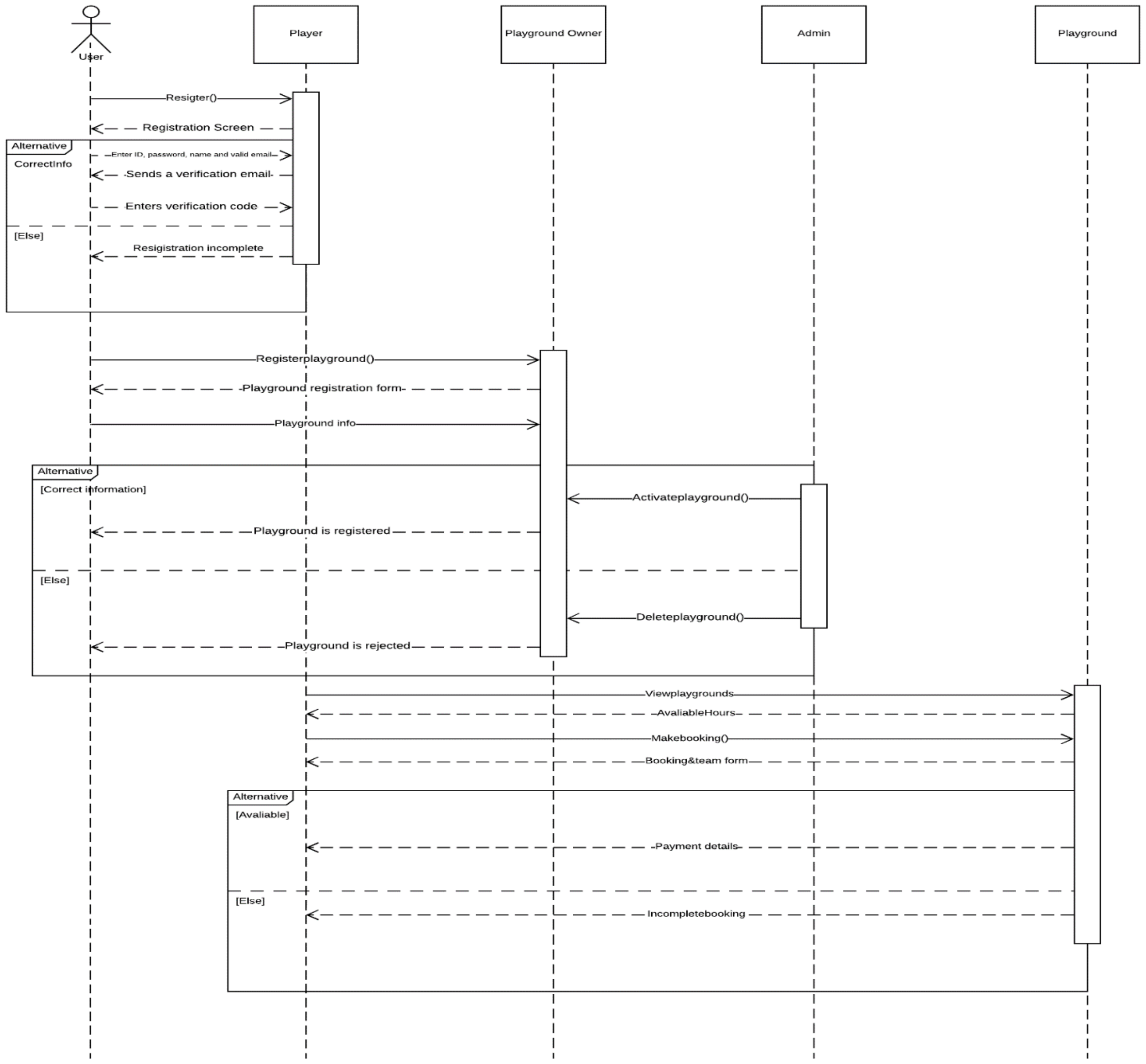
II. Class Descriptions

Class ID	Class Name	Description & Responsibility
1.	Player	• Register a user profile. • Login to the system. • Book playground. • Pay playground fees.
2.	Admin	• Register a user profile. • Login to the system. • Activate playgrounds. • Suspend playgrounds. • Delete playgrounds.
3.	Playground Owner	• Register a user profile. • Login to the system. • Register playgrounds.
4.	Booking	• Register booking information. • Pay playground fees. • Cancel playground booking.
5.	Playground	• Register playground information. • Confirm Booking.
6.	Electronic E-Wallet	• Responsible for finance.

CS251: Phase 2 – <Tankers>
Project: <GoFo>

Software Design Specification

III. Sequence diagrams



Software Design Specification

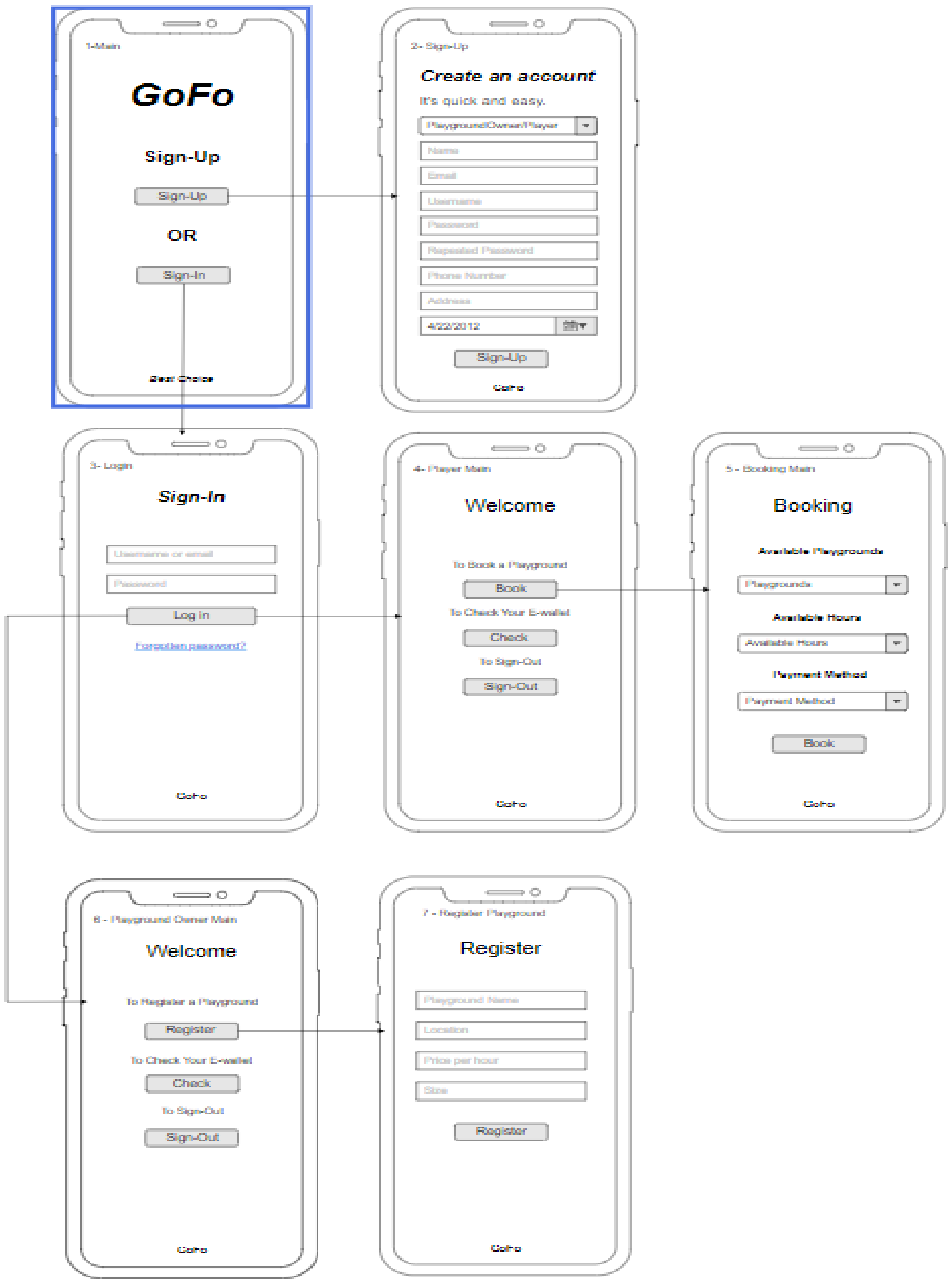
Class - Sequence Usage Table

Sequence Diagram	Classes Used	All Methods Used
1. Register User	Class Player	Register ()
2. Add a Playground & Approve Playground	Class Playground Owner Class Admin	RegisterPlayground () ActivePlayground ()
3. Book a Playground & View available Hours	Class Player Class Playground	MakeBooking ()

CS251: Phase 2 – <Tankers>
Project: <GoFo>

Software Design Specification

IV. User Interface Design



Software Design Specification

Screen ID	Screen Name	Screen / Wireframe Description
1.	Main	Enables users to register and login.
2.	Sign-Up	Enables users to create an account and a profile.
3.	Login	Enables users to access their accounts.
4.	Player Main	Enables players to perform his/her operations.
5.	Booking Main	Enables player to book a playground.
6.	Playground Owners Main	Enables Playground owners to perform his/her operations.
7.	Register Playground	Enables Playground owners to register a playground.

• Tools

- Lucid Chart Website.
- Moqups Website.
- Eclipse JAVA IDE.

• Ownership Report

Item	Owners
Seif Mosaad Abd El -Fattah	Implementation / Git.
Ahmed Nabil Mohamed Salah	Implementation / Git.
Habiba Amr Mohamed	Ui / Part from Sequence Diagram.
Marina Moheb Nafee	Part from Sequence Diagram / Class Diagram.

Software Design Specification

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
- www.lucidchart.com
- https://stackoverflow.com/

Appendix A: Code Listing and Screen Snapshots

Admin.java

```
package model;



import java.io.Serializable;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;

import java.util.Map.Entry;



public class Admin extends Player implements Serializable{



    public void Role(HashMap<Integer, Playground> s)

    {

        Scanner in = new Scanner(System.in);
```

Software Design Specification

```
while(true) {

   System.out.println("1- To delete a playground");

   System.out.println("2- To suspend a playground");

   System.out.println("3- To active a playground");

   System.out.println("4- To back to main menu");

   System.out.print("Please enter your choice : ");

   int cho = in.nextInt();

   if(cho == 1)

   {

      for (Entry<Integer, Playground> mapElement : s.entrySet())

      {



         System.out.println("Playground id ["+mapElement.getValue().getId()+"]
"+mapElement.getValue().getName()+" "+mapElement.getValue().getLocation()+"
"+mapElement.getValue().getPricePerHour()+" "+mapElement.getValue().getSize()+"
"+mapElement.getValue().getStatus());

      }

      System.out.print("Please enter playground's id : ");

      int c = in.nextInt();

      if(deletePlayground(s, c))

      {System.out.print("The playground was successfully removed to back to main menu 4 or 0 to do
another operation : ");}

      else {
```

```
                    System.out.print("The playground was unsuccessfully removed to back to main menu 4 or 0 to do
another operation : ");

                }

                c = in.nextInt();

                if(c==4)

                    break;

                else if(c==0)

                    continue;

            }

            else if(cho==2)

            {

                for (Entry<Integer, Playground> mapElement : s.entrySet())

                {

                    System.out.println("Playground id ["+mapElement.getValue().getId()+"]
"+mapElement.getValue().getName()+" "+mapElement.getValue().getLocation()+"
"+mapElement.getValue().getPricePerHour()+" "+mapElement.getValue().getSize()+"
"+mapElement.getValue().getStatus());

                }

                System.out.print("Please enter playground's id : ");

                int c = in.nextInt();

                if(suspendPlayground(s, c))

                {System.out.print("The playground was successfully suspended to back to main menu 4 or 0 to do
another operation : ");}

                else {
```

```
                System.out.print("The playground was unsuccessfully suspended to back to main menu 4 or 0 to
do another operation : ");

            }

            c = in.nextInt();

            if(c==4)

                break;

            else if(c==0)

                continue;

        }

        else if(cho==3)

        {

            for (Entry<Integer, Playground> mapElement : s.entrySet())

            {

                System.out.println("Playground id ["+mapElement.getValue().getId()+"]
"+mapElement.getValue().getName()+" "+mapElement.getValue().getLocation()+"
"+mapElement.getValue().getPricePerHour()+" "+mapElement.getValue().getSize()+"
"+mapElement.getValue().getStatus());

            }

            System.out.print("Please enter playground's id : ");

            int c = in.nextInt();

            if(activePlayground(s, c))

            {System.out.print("The playground was successfully activated to back to main menu 4 or 0 to do
another operation : ");}

            else {
                System.out.print("The playground was unsuccessfully activated to back to main menu 4 or 0 to do
another operation : ");
            }
            c = in.nextInt();
            if(c==4)
                break;
            else if(c==0)
                continue;
        }
        else if(cho==4)
        {
            break;
        }
        else {
            continue;
        }
    }
    return;
}


public boolean deletePlayground(HashMap<Integer, Playground> s , int id)
{
    if(s.containsKey(id))
    {
        s.remove(id);
        return true;
    }
    else {
        return false;
    }
}
public boolean suspendPlayground(HashMap<Integer, Playground> s , int id)
{
    if(s.containsKey(id))
    {
        s.get(id).setStatus(0);
        return true;
    }
    else {
        return false;
    }
}
public boolean activePlayground(HashMap<Integer, Playground> s , int id)
{
      if(s.containsKey(id))

      {

         s.get(id).setStatus(2);

         return true;

      }

      else {

         return false;

      }

   }

}
```

Booking.java

```
package model;

import java.io.Serializable;

public class Booking implements Serializable{
    int id;
    Player bookingPlayer;
    Date date;
    int from;
    int to;
    int status;

    public Booking(int id, Player bookingPlayer, Date date, int from, int to, int
status) {
        this.id = id;
        this.bookingPlayer = bookingPlayer;
        this.date = date;
        this.from = from;
        this.to = to;
        this.status = status;
    }
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public Player getBookingPlayer() {
        return bookingPlayer;
    }

    public void setBookingPlayer(Player bookingPlayer) {
        this.bookingPlayer = bookingPlayer;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

    public int getFrom() {
        return from;
    }

    public void setFrom(int from) {
        this.from = from;
    }

    public int getTo() {
        return to;
    }

    public void setTo(int to) {
        this.to = to;
    }

    public int getStatus() {
        return status;
    }

    public void setStatus(int status) {
        this.status = status;
    }

    boolean cancel(){
        return true;
    }
    boolean pay(){
        return true;
    }
}
```

Data.java

```
package model;



import java.io.Serializable;

import java.util.Calendar;



public class Date implements Serializable{

  private static final String weekDays[] = {"Saturday", "Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday"};

  private static final int monthDays[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

  private int year;

  private int month;

  private int day;

  public Date(){

    year = 1920; month = 1; day = 1;

  }

  public Date(int day, int month, int year){
    this.year = year; this.month = month; this.day = day;

  }


  public int getYear() {

    return year;

  }

  public void setYear(int year) {

    this.year = year;

  }


  public int getMonth() {

    return month;

  }

  public void setMonth(int month) {

    this.month = month;

  }


  public int getDay() {

    return day;

  }

  public void setDay(int day) {

    this.day = day;
```

```
}


// REQUIRES: A valid date
// EFFECTS: returns the day name of the current date
public String getDayName(){
    int years = year - 1920, leapYears = years / 4, normalYears = years - leapYears;
    int days = normalYears * 365 + leapYears * 366 + 4 + day;
    boolean leapYear = years % 4 == 0;
    for (int i = 0; i < month - 1; ++i) {
        days += monthDays[i];
        if (i == 1 && leapYear) days++;
    }
    return weekDays[days % 7];
}
// REQUIRES: A valid date
// EFFECTS: returns the date in format "DD-MM-YYYY"
public String getDate(){
    return (day + "-" + month + "-" + year);
    //  + ": " + getDayName() + " February 25, 2020");
}


// EFFECTS: returns true if the data is valid else false
```

```
   public boolean valid(){

      int year = Calendar.getInstance().get(Calendar.YEAR), month =
Calendar.getInstance().get(Calendar.MONTH) + 1, day =
Calendar.getInstance().get(Calendar.DAY_OF_MONTH);

      if (this.year >= year && this.month >=  month && this.day >= day && this.month > 0 && this.month <=
12 &&

            this.day > 0 && this.day <= monthDays[this.month - 1])

         return true;

      return false;

   }


   // EFFECTS: returns true if current date is today

   public boolean today(){

      if (this.year == Calendar.getInstance().get(Calendar.YEAR) && this.month ==
Calendar.getInstance().get(Calendar.MONTH) + 1 &&

            this.day == Calendar.getInstance().get(Calendar.DAY_OF_MONTH))

         return true;

      return false;

   }

}
```

Software Design Specification

Player.java

```
package model;


import java.io.Serializable;

import java.util.HashMap;

import java.util.List;

import java.util.Map.Entry;

import java.util.Scanner;


public class Player implements Serializable{

    protected int id;

    protected String name;

    protected String dob;

    protected String address;

    protected String mobile;

    protected String email;

    protected String username;

    protected String password;


    public Player()

    {

        this.id = 0;
        this.name="";

        this.dob = "";

        this.address="";

        this.mobile ="";

        this.email="";

        this.username="";

        this.password="";

    }

    public Player(int id , String name , String dob , String address , String mobile , String email , String
username , String password)

    {

        this.id = id;

        this.name=name;

        this.dob = dob;

        this.address=address;

        this.mobile =mobile;

        this.email=email;

        this.username=username;

        this.password=password;

    }


    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDob() {
        return dob;
    }
    public void setDob(String dob) {
        this.dob = dob;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
```

```
}

public String getMobile() {

   return mobile;

}

public void setMobile(String mobile) {

   this.mobile = mobile;

}

public String getEmail() {

   return email;

}

public void setEmail(String email) {

   this.email = email;

}

public String getUsername() {

   return username;

}

public void setUsername(String username) {

   this.username = username;

}

public String getPassword() {

   return password;

}
```

```
   public void setPassword(String password) {

      this.password = password;

   }


   public void Role(HashMap<Integer, Playground> s)

   {

      Scanner in = new Scanner(System.in);

      if(s.isEmpty())

      {

          System.out.println("There Aren't Any Playgrounds To Book");

          return;

      }

      for (Entry<Integer, Playground> mapElement : s.entrySet())

      {

         if(mapElement.getValue().getStatus()==2)

         {System.out.println("Playground id ["+mapElement.getValue().getId()+"]
"+mapElement.getValue().getName()+" "+mapElement.getValue().getLocation()+"
"+mapElement.getValue().getPricePerHour()+" "+mapElement.getValue().getSize());}

      }

      System.out.print("Please enter playground's id : ");

      int c = in.nextInt();

      if(s.containsKey(c) && s.get(c).getStatus() == 2)

      {
```

```
Date date;

int day , month , year ,from , to, i = 0;

do {

   if (i++ > 0) System.out.println("Invalid date!! Please Enter a valid date:");

   System.out.print("Please enter the year : ");

   year = in.nextInt();


   System.out.print("Please enter the month : ");

   month = in.nextInt();


   System.out.print("Please enter the day : ");

   day = in.nextInt();


   date = new Date(day, month, year);

} while(!date.valid());

i = 0; boolean foundFrom, foundTo;

do {

   if (i++ > 0) System.out.println("Invalid times!");

   List<Integer> availableHours = s.get(c).showAvailableHours(date);

   foundFrom = false; foundTo = false;

   System.out.println("***********Avaliable Hours***********");

   System.out.println(availableHours);
```

```
            System.out.print("Please enter the time from : ");

            from = in.nextInt();

            for (Integer hour : availableHours) if (hour == from) {foundFrom = true; break;}

            System.out.print("Please enter the time to : ");

            to = in.nextInt();

            for (Integer hour : availableHours) if (hour == to) {foundTo = true; break;}

          } while(from > to || !foundFrom || !foundTo);

          s.get(c).makeBooking(this, date, from, to);

          System.out.println("Booking was succesfull");

      }

      //in.close();

    }

    @Override

    public String toString() {

        return "Player [id=" + id + ", name=" + name + ", dob=" + dob + ", address=" + address + ", mobile=" +
mobile

            + ", email=" + email + ", username=" + username + ", password=" + password + "]";

    }

}
```

Software Design Specification

Players.java

```
package model;


import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

import java.text.ParsePosition;

import java.text.SimpleDateFormat;

import java.util.HashMap;

import java.util.Scanner;


public class Players  implements Serializable{

    public HashMap<String, Player> users = new HashMap<String, Player>();

    public int ih = 1;

    public boolean Register(Player p)

    {

        Scanner in = new Scanner(System.in);

        String name ; String dob ; String address ; String mobile ; String email ; String username ; String
password;
```

```
p.id = this.ih;

System.out.print("Please enter your name : ");

name = in.nextLine();

System.out.print("Please enter your dob : ");

dob = in.nextLine();

System.out.print("Please enter your address : ");

address = in.nextLine();

System.out.print("Please enter your mobile : ");

mobile = in.nextLine();

System.out.print("Please enter your email : ");

email = in.nextLine();

System.out.print("Please enter your username : ");

username = in.nextLine();

System.out.print("Please enter your password : ");

password = in.nextLine();

while(true)

{

  if(name.isEmpty())

  {

    System.out.println("Name field can not be empty please re-enter it");

    System.out.print("Please enter your name : ");

    name = in.nextLine();
}
else if(dob.isEmpty() || !isDateValid(dob))
{
   System.out.println("Date of birth field can not be empty, or wrong data form please re-enter it");
   System.out.print("Please enter your dob : ");
   dob = in.next();
   in.nextLine();
}
else if(address.isEmpty())
{
   System.out.println("Address field can not be empty please re-enter it");
   System.out.print("Please enter your address : ");
   address = in.nextLine();


}
else if(!validatePhoneNumber(mobile))
{
   System.out.println("mobile field can not be empty or mobile email form please re-enter it");
   System.out.print("Please enter your mobile : ");
   mobile = in.nextLine();
}
else if(!isValid(email))
```

```
{
   System.out.println("Email field can not be empty or wronge email form please re-enter it");

   System.out.print("Please enter your email : ");

   email = in.nextLine();

}

else if(username.isEmpty())

{


   System.out.println("Username field can not be empty please re-enter it");

   System.out.print("Please enter your username : ");

   username = in.nextLine();

}

else if(users.containsKey(username))

{


   System.out.println("Username is taken please enter another one");

   System.out.print("Please enter your username : ");

   username = in.nextLine();

}

else if(password.isEmpty())

{
   System.out.println("Username field can not be empty please re-enter it");

   System.out.print("Please enter your password : ");

   password = in.nextLine();

}

else {

   p.name = name;

   p.dob = dob;

   p.address=address;

   p.mobile=mobile;

   email.trim();

   email = email.replaceAll("\\s","");

   p.email=email;

   username.trim();

   username = username.replaceAll("\\s","");

   username.toLowerCase();

   p.username=username;

   p.password=password;

   this.ih++;

   users.put(p.username, p);

   //in.close();

   return true;

}
```

```
        }
    }


    public int Login(String un , String pwd)
    {
        un.trim();
        un = un.replaceAll("\\s","");
        un.toLowerCase();
        int back = 0;
        if(!users.containsKey(un))
            return back;
        if(users.containsKey(un) && !users.get(un).password.equals(pwd))
            back = 1;
        if(users.containsKey(un) && users.get(un).password.equals(pwd))
        {
            back = 2;
        }

        return back;
    }

    public  boolean validatePhoneNumber(String phoneNo) {
```

```
    //validate phone numbers of format "1234567890"
    if (phoneNo.matches("\\d{10}")) return true;
      //validating phone number with -, . or spaces
    else if(phoneNo.matches("\\d{3}[-\\.\\s]\\d{3}[-\\.\\s]\\d{4}")) return true;
      //validating phone number with extension length from 3 to 5
    else if(phoneNo.matches("\\d{3}-\\d{3}-\\d{4}\\s(x|(ext))\\d{3,5}")) return true;
      //validating phone number where area code is in braces ()
    else if(phoneNo.matches("\\(\\d{3}\\)-\\d{3}-\\d{4}")) return true;
      //return false if nothing matches the input
    else return false;


  }
  public boolean isValid(String email) {
    email.trim();
    email = email.replaceAll("\\s","");
    String regex = "^[\\w-_\\.+]*[\\w-_\\.]\\@([\\w]+\\.)+[\\w]+[\\w]$";
    return email.matches(regex);
  }
  public void save ()
  {
    try
    {
            FileOutputStream fos =
                new FileOutputStream("hashusers.txt");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(users);
            oos.close();
            fos.close();
        }catch(IOException ioe)
        {
            ioe.printStackTrace();
        }
    }
    public void load()
    {
        try
        {
            FileInputStream fis = new FileInputStream("hashusers.txt");
            ObjectInputStream ois = new ObjectInputStream(fis);
            users = (HashMap) ois.readObject();
            ois.close();
            fis.close();
        }catch(IOException ioe)
        {
```

```
        Player adPlayer = new Admin();

        adPlayer.setId(-1);

        adPlayer.setName("admin");

        adPlayer.setDob("02-02-2020");

        adPlayer.setAddress("19-Admin");

        adPlayer.setMobile("1234567895");

        String emailS = "admin@gmail.com";

        emailS.trim();

        emailS = emailS.replaceAll("\\s","");

        adPlayer.setEmail(emailS);

        String usernameS = "admin";

        usernameS.trim();

        usernameS = usernameS.replaceAll("\\s","");

        adPlayer.setUsername(usernameS);

        adPlayer.setPassword("admin");

        users.put(adPlayer.username,adPlayer);

        return;

    }catch(ClassNotFoundException c)

    {

        System.out.println("Class not found");

        c.printStackTrace();

        return;
        }

    }

    public static boolean isDateValid(String s)

    {

        SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");

        sdf.setLenient(false);

        return sdf.parse(s, new ParsePosition(0)) != null;

    }

}
```

Playground.java

```
package model;



import java.io.Serializable;

import java.util.*;



import model.Date;

import model.Booking;



public class Playground implements Serializable {

    public static int bookingId = 0;

    public static long playgroundId = 0;
    private long id;

    private String name;

    private String location;

    private Double size;

    private int pricePerHour;

    private int cancellationPeriod;

    private HashMap<String, ArrayList<Integer>> availableHours;

    private int status;

    private List<Booking> bookings;


    // constructors

    public Playground(){

        this.id = playgroundId++;

        availableHours = new HashMap<String, ArrayList<Integer>>();

        bookings = new ArrayList<Booking>();

        this.status = 1;

    }

    public Playground(String name, String location, Double size, int pricePerHour, int cancellationPeriod, int
status){

        this.name = name; this.location = location; this.size = size; this.pricePerHour = pricePerHour;

        this.cancellationPeriod = cancellationPeriod; this.status = status;
        this.id = playgroundId++;

        availableHours = new HashMap<String, ArrayList<Integer>>();

        bookings = new ArrayList<Booking>();

    }


    // setters & getters

    public long getId() {

        return id;

    }

    public void setId(long id) {

        this.id = id;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getLocation() {

        return location;

    }
```

```
  public void setLocation(String location) {
     this.location = location;
  }
  public Double getSize() {
     return size;
  }
  public void setSize(Double size) {
     this.size = size;
  }
  public int getPricePerHour() {
     return pricePerHour;
  }
  public void setPricePerHour(int pricePerHour) {
     this.pricePerHour = pricePerHour;
  }
  public int getCancellationPeriod() {
     return cancellationPeriod;
  }
  public void setCancellationPeriod(int cancellationPeriod) {
     this.cancellationPeriod = cancellationPeriod;
  }
  public HashMap<String, ArrayList<Integer>> getAvailableHours() {
    return new HashMap<String, ArrayList<Integer>>(availableHours);

  }

  public void setAvailableHours(HashMap<String, ArrayList<Integer>> availableHours) {

    this.availableHours = availableHours;

  }

  public int getStatus() {

    return status;

  }

  public void setStatus(int status) {

    this.status = status;

  }

  public List<Booking> getBookings() {

    return new ArrayList<Booking>(bookings);

  }

  public void setBookings(List<Booking> bookings) {

    this.bookings = bookings;

  }




  // EFFECTS: returns a list of available hours for the given date

  public List<Integer> showAvailableHours(Date date){

    List<Integer> ret = new ArrayList<Integer>();
```

```
        if (availableHours.containsKey(date.getDate())) ret = availableHours.get(date.getDate());
        else for (int i = 0; i < 24; ++i) ret.add(i);
        int cur_hr = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
        if (date.today())
            for (int i = 0; i < ret.size(); ++i) if (ret.get(i) <= cur_hr) ret.remove(i--);
        return new ArrayList<Integer>(ret);
    }


    // REQUIRES: Given date and time are available
    // MODIFIES: this
    // EFFECTS: adds a new booking in bookings, returns true if operation was successful
    public boolean makeBooking(Player p, Date date, int from, int to){
        Booking booking = new Booking(bookingId++, p, date, from, to, status);
        bookings.add(booking);
        if (!availableHours.containsKey(date.getDate())) {
            availableHours.put(date.getDate(), new ArrayList<Integer>(
                    Arrays.asList(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 ,16, 17, 18, 19, 20, 21, 22, 23)));
        }
        List<Integer> list = availableHours.get(date.getDate());
        for (int i = 0; i < list.size(); ++i) if (list.get(i) >= from && list.get(i) <= to) list.remove(i--);


        return true;
    }


    // REQUIRES: Given id has to exist in bookings
    // MODIFIES: this
    // EFFECTS: removes a booking from bookings returns true if operation is successful
    public boolean cancelBooking(int bid){
        int i;
        for (i = 0; i < bookings.size(); ++i)
            if (bookings.get(i).getId() == bid) {  break; }

        List<Integer> list = availableHours.get(bookings.get(i).getDate().getDate());
        for(int t = bookings.get(i).getFrom(); t <= bookings.get(i).getTo(); ++t) list.add(t);
        Collections.sort(list);
        bookings.remove(i);

        return true;
    }
}
```

Software Design Specification

PlaygroundOwner.java

```
package model;


import java.io.Serializable;

import java.util.HashMap;

import java.util.Scanner;


import model.Player;

import model.Playground;

public class PlaygroundOwner extends Player implements Serializable {


  public void Role(HashMap<Integer, Playground> s)

  {

    Scanner in = new Scanner(System.in);

    while(true) {

      System.out.println("1- Resigt playground");

      System.out.println("2- back to main menu");

      System.out.print("Please enter your choice : ");

      int cho = in.nextInt();

      if(cho == 1)

      {

        in.nextLine();
```

```
            System.out.print("Please enter the name : ");

            String name1 = in.nextLine();

            System.out.print("Please enter the location : ");

            String location = in.nextLine();

            System.out.print("Please enter the Price Per Hour : ");

            int pricePerHour = in.nextInt();

            System.out.print("Please enter the size : ");

            Double size = in.nextDouble();

            Playground p = new Playground();

            p.setName(name1);

            p.setLocation(location);

            p.setPricePerHour(pricePerHour);

            p.setSize(size);

            s.put((int) p.getId(), p);

            System.out.print("Registering is succesful to back to main menu 4 or 0 to do another operation : ");

            int c1 = in.nextInt();

            if(c1==4)

            {break;

            }

            else

               continue;

         }
```

```
            else if (cho==2)
                break;
            else
                continue;
        }
        //in.close();
        return;
    }
}
```

Main.java

```
package ui;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;


import model.Admin;
import model.Player;
import model.Players;
import model.Playground;
import model.PlaygroundOwner;
```

Software Design Specification

```
public class Main {


    public static void main(String[] args) {

        HashMap<Integer, Playground>  playgrounds = new HashMap<Integer, Playground>();

        Players players = new Players();

        players.load();

        int cho;

        Scanner in1 = new Scanner(System.in);

        do {

            System.out.println("Welcome to GOFO Booking System");

            System.out.println("1- Login");

            System.out.println("2- Resigter");

            System.out.println("3- Exit");

            System.out.print("Please enter your choice : ");

            cho = in1.nextInt();

            if(cho == 1)

            {

                in1.nextLine();

                while(true)

                {

                    String un , pwd ;

                    System.out.println("Please enter your username : ");
```

```
un = in1.nextLine();
un.trim();
un = un.replaceAll("\\s","");
System.out.println("Please enter your password : ");
pwd = in1.nextLine();
int result =players.Login(un, pwd);
if(result==0)
{
  int u ;
  System.out.print("Username is not existed, to re-login press 1 or 0 to back to main menu : ");
  u = in1.nextInt();
  in1.nextLine();
  if(u == 1)
    continue;
  else if(u == 0)
    break;
}
else if(result==1)
{
  int u ;
  System.out.print("Username or Password is not correct, to re-login press 1 or 0 to back to main
menu : ");
```

```
                        u = in1.nextInt();
                        in1.nextLine();
                        if(u == 1)
                            continue;
                        else if(u == 0)
                            break;
                    }
                    else {
                        System.out.println("Welcome , "+players.users.get(un).getName());
                        players.users.get(un).Role(playgrounds);
                        break;
                    }
                }
                continue;
            }
            else if(cho == 2)
            {
                in1.nextLine();
                System.out.print("Please enter your role [p/pg] : ");
                String c = in1.nextLine();
                c.toLowerCase();
                if(c.equals("p"))
```

```
    {
        Player player = new Player();
        if(players.Register(player))
        {
            System.out.println("Register was successful");
            continue;
        }
        else
        {
            System.out.println("Register was unsuccessful");
            continue;
        }
    }
    else if(c.equals("pg"))
    {
        Player playgroundOwner = new PlaygroundOwner();
        if(players.Register(playgroundOwner))
        {
            System.out.println("Register was successful");
            continue;
        }
        else
```

```
                {
                    System.out.println("Register was unsuccessful");
                    continue;
                }
            }
            else {
                System.out.println("Wrong Choice");
                continue;
            }
        }
        else if(cho==3)
        {
            players.save();
            break;
        }
    }while(cho !=3);
  }

}
```

PlaygroundTest.java

```
package test;



import model.Date;

import model.Player;

import model.Playground;



import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import org.junit.Before;

import org.junit.Test;


import static junit.framework.TestCase.assertEquals;


public class PlaygroundTest {

    String testName;

    String testLocation;

    Double testSize;

    int testPricePerHour;
    int testCancellationPeriod;

    int testStatus;

    Player testPlayer;

    Date testDate;

    Playground testPlayground;

    @Before

    public void setUp(){

        testName = "test name";

        testLocation = "test street, testier governorate, testiest city ";

        testSize = 100.15;

        testPricePerHour = 10;

        testCancellationPeriod = 10;

        testStatus = 4;

        testPlayer = new Player();

        testDate = new Date(10, 11, 2300);

        testPlayground = new Playground(testName, testLocation, testSize, testPricePerHour,
testCancellationPeriod, testStatus);

    }


    @Test

    public void testConstructors(){

        assertEquals(Playground.playgroundId - 1, testPlayground.getId());
        assertEquals(testName, testPlayground.getName());

        assertEquals(testLocation, testPlayground.getLocation());

        assertEquals(testSize, testPlayground.getSize());

        assertEquals(testPricePerHour, testPlayground.getPricePerHour());

        assertEquals(testCancellationPeriod, testPlayground.getCancellationPeriod());

        assertEquals(testStatus, testPlayground.getStatus());

        assertEquals(true, testPlayground.getAvailableHours().isEmpty());

        assertEquals(true, testPlayground.getBookings().isEmpty());


        Playground p2 = new Playground();

        assertEquals(Playground.playgroundId - 1, p2.getId());

    }


    @Test

    public void testShowAvailableHours(){

        List<Integer> testAvailableHours = new ArrayList<Integer>(

                Arrays.asList(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23));


        assertEquals(true, testAvailableHours.equals(testPlayground.showAvailableHours(testDate)));

    }


    @Test
```

```
public void testMakeBookingEmptyDate(){
    testPlayground.makeBooking(testPlayer, testDate, 12, 14);
    int testValue = -1;
    if (!testPlayground.getBookings().isEmpty()) testValue = testPlayground.getBookings().get(0).getId();
    assertEquals(Playground.bookingId - 1, testValue);
    List<Integer> testAvailableHours = new ArrayList<Integer>(
            Arrays.asList(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 15, 16, 17, 18, 19, 20, 21, 22, 23));

    assertEquals(true, testAvailableHours.equals(testPlayground.showAvailableHours(testDate)));
}


@Test
public void testMakeBooking(){
    testPlayground.makeBooking(testPlayer, testDate, 12, 14);
    testPlayground.makeBooking(testPlayer, testDate, 9, 10);
    int testValue = -1;
    if (!testPlayground.getBookings().isEmpty()) testValue = testPlayground.getBookings().get(0).getId();
    assertEquals(Playground.bookingId - 2, testValue);

    testValue = -1;
    if (!testPlayground.getBookings().isEmpty()) testValue = testPlayground.getBookings().get(1).getId();
    assertEquals(Playground.bookingId - 1, testValue);
```

```
        List<Integer> testAvailableHours = new ArrayList<Integer>(
                Arrays.asList(0, 1, 2, 3, 4, 5, 6, 7, 8, 11, 15, 16, 17, 18, 19, 20, 21, 22, 23));

        assertEquals(true, testAvailableHours.equals(testPlayground.showAvailableHours(testDate)));

        testPlayground.makeBooking(testPlayer, testDate, 22, 23);
        testValue = -1;
        if (!testPlayground.getBookings().isEmpty()) testValue = testPlayground.getBookings().get(2).getId();
        assertEquals(Playground.bookingId - 1, testValue);

        testAvailableHours = new ArrayList<Integer>(
                Arrays.asList(0, 1, 2, 3, 4, 5, 6, 7, 8, 11, 15, 16, 17, 18, 19, 20, 21));

        assertEquals(true, testAvailableHours.equals(testPlayground.showAvailableHours(testDate)));
    }

    @Test
    public void testCancelBooking(){
        testPlayground.makeBooking(testPlayer, testDate, 12, 14);
        testPlayground.makeBooking(testPlayer, testDate, 9, 10);
        assertEquals(testPlayground.getBookings().size(), 2);
        testPlayground.cancelBooking(0);

        assertEquals(testPlayground.getBookings().size(), 1);

        int testValue = -1;

        if (!testPlayground.getBookings().isEmpty()) testValue = testPlayground.getBookings().get(0).getId();

        assertEquals(Playground.bookingId - 1, testValue);


        List<Integer> testAvailableHours = new ArrayList<Integer>(

                Arrays.asList(0, 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23));

        assertEquals(true, testAvailableHours.equals(testPlayground.showAvailableHours(testDate)));

    }

}
```

GitHub Information:

Repo link: https://github.com/SeifMosad/GoFo

Repo Path : https://github.com/SeifMosad/GoFo.git

GitHub Login:

Username: SeifMosad

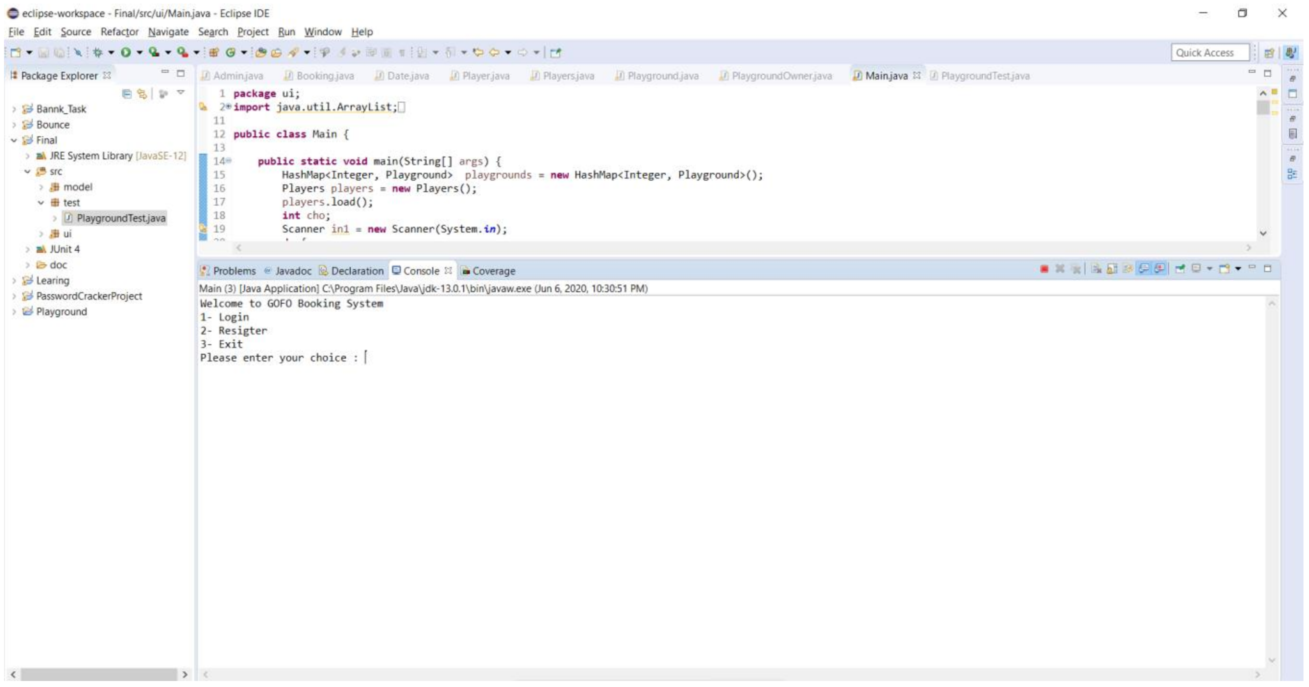
Password: admin2000fci

Google Link: https://drive.google.com/drive/folders/1WKO0BNk0mEbCw5tH-mXrHdb0e_wHAIol?usp=sharing

CS251: Phase 2 – <Tankers>
Project: <GoFo>

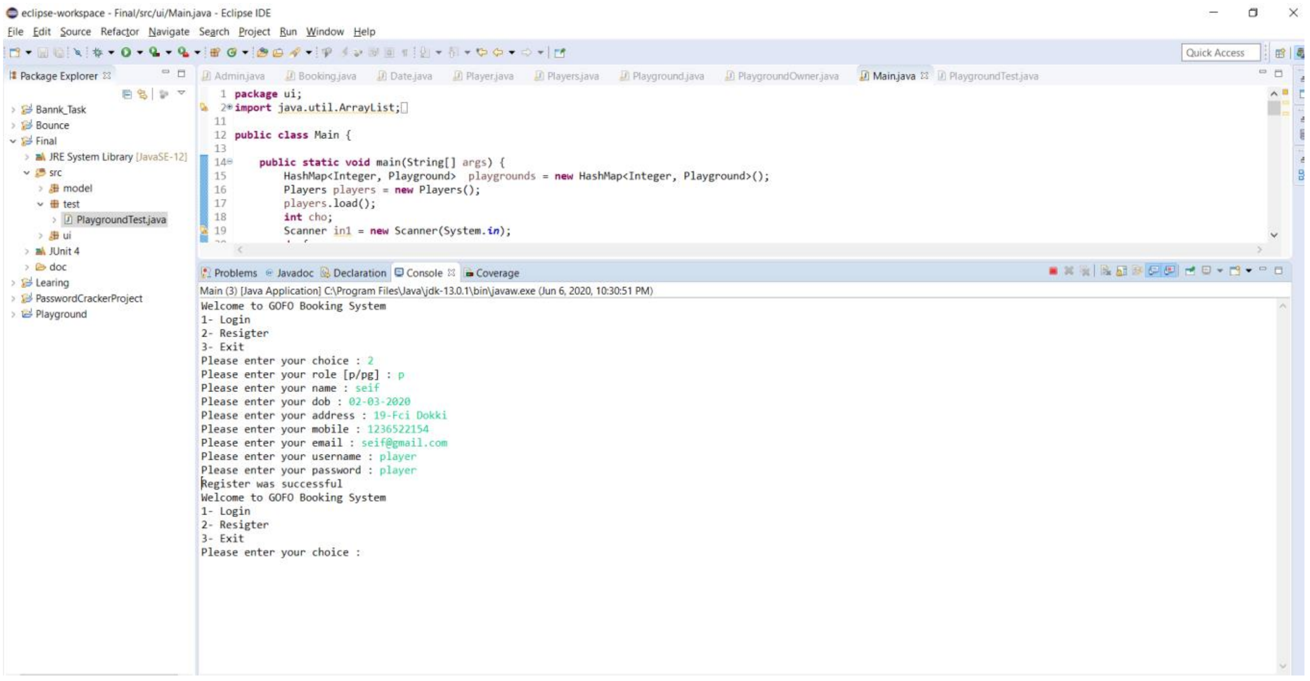
Software Design Specification

Screen Shots


eclipse-workspace - Final/src/ui/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Bannk_Task
Bounce
Final
JRE System Library [JavaSE-12]
src
model
test
PlaygroundTest.java
ui
JUnit 4
doc
Learing
PasswordCrackerProject
Playground
Admin.java Booking.java Date.java Player.java Players.java Playground.java PlaygroundOwner.java Main.java PlaygroundTest.java
Problems Javadoc Declaration Console Coverage

```
package ui;
import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        HashMap<Integer, Playground>  playgrounds = new HashMap<Integer, Playground>();
        Players players = new Players();
        players.load();
        int cho;
        Scanner in1 = new Scanner(System.in);
```

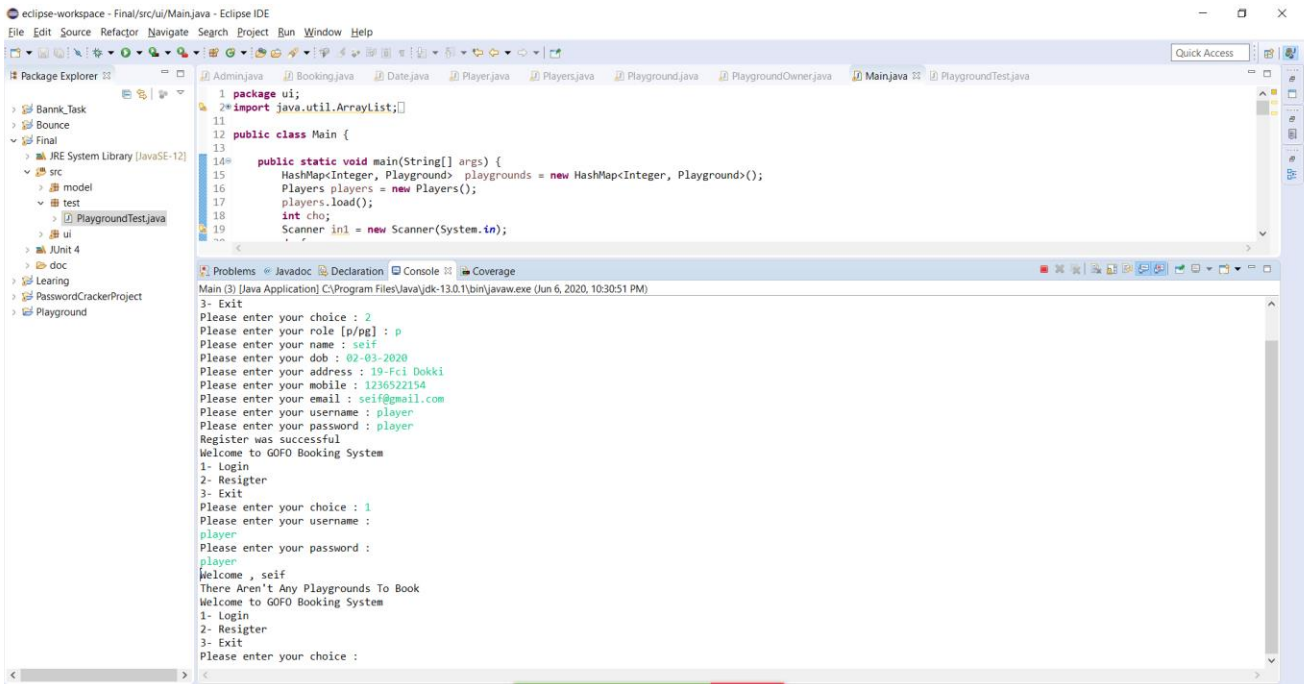
```
Main (3) [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Jun 6, 2020, 10:30:51 PM)
Welcome to GOFO Booking System
1- Login
2- Resigter
3- Exit
Please enter your choice :
```


eclipse-workspace - Final/src/ui/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Bannk_Task
Bounce
Final
JRE System Library [JavaSE-12]
src
model
test
PlaygroundTest.java
ui
JUnit 4
doc
Learing
PasswordCrackerProject
Playground
Admin.java Booking.java Date.java Player.java Players.java Playground.java PlaygroundOwner.java Main.java PlaygroundTest.java
Problems Javadoc Declaration Console Coverage

```
package ui;
import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        HashMap<Integer, Playground>  playgrounds = new HashMap<Integer, Playground>();
        Players players = new Players();
        players.load();
        int cho;
        Scanner in1 = new Scanner(System.in);
```

```
Main (3) [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Jun 6, 2020, 10:30:51 PM)
Welcome to GOFO Booking System
1- Login
2- Resigter
3- Exit
Please enter your choice : 2
Please enter your role [p/pg] : p
Please enter your name : seif
Please enter your dob : 02-03-2020
Please enter your address : 19-Fci Dokki
Please enter your mobile : 1236522154
Please enter your email : seif@gmail.com
Please enter your username : player
Please enter your password : player
Register was successful
Welcome to GOFO Booking System
1- Login
2- Resigter
3- Exit
Please enter your choice :
```

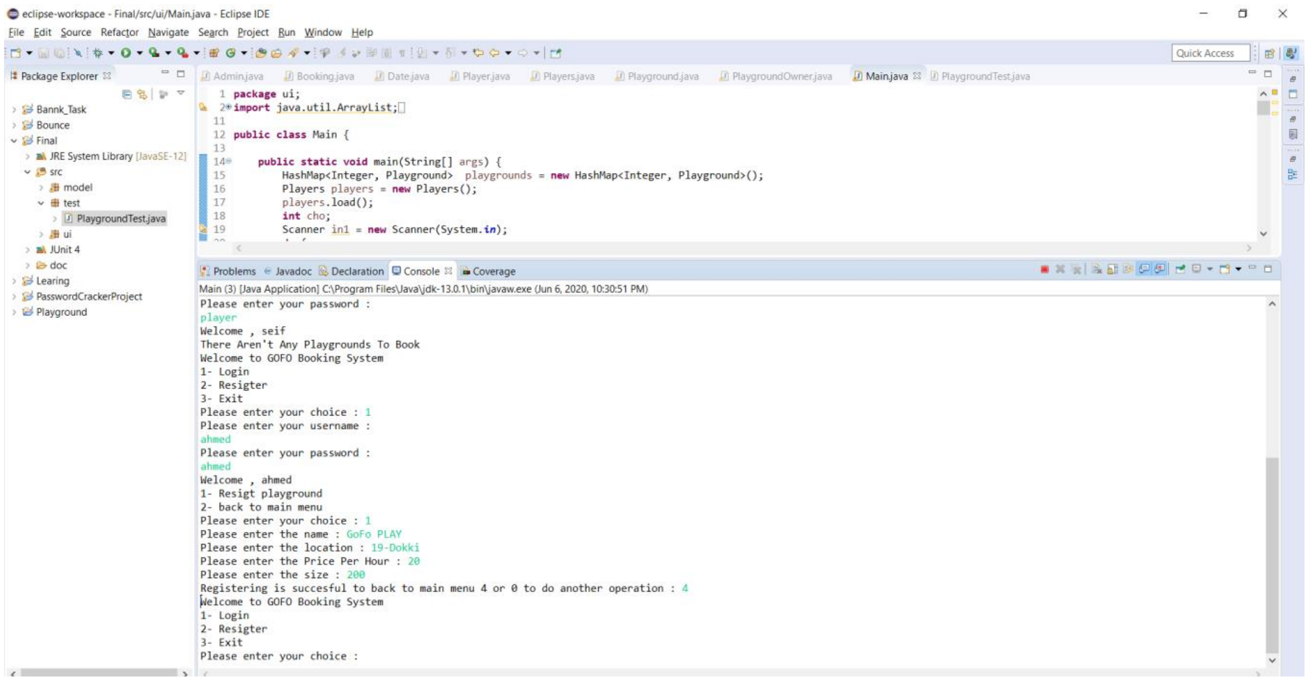
Software Design Specification



```
package ui;
import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        HashMap<Integer, Playground>  playgrounds = new HashMap<Integer, Playground>();
        Players players = new Players();
        players.load();
        int cho;
        Scanner in1 = new Scanner(System.in);
```

Main (3) [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Jun 6, 2020, 10:30:51 PM)

```
3- Exit
Please enter your choice : 2
Please enter your role [p/pg] : p
Please enter your name : seif
Please enter your dob : 02-03-2020
Please enter your address : 19-Fci Dokki
Please enter your mobile : 1236522154
Please enter your email : seif@gmail.com
Please enter your username : player
Please enter your password : player
Register was successful
Welcome to GOFO Booking System
1- Login
2- Resigter
3- Exit
Please enter your choice : 1
Please enter your username :
player
Please enter your password :
player
Welcome , seif
There Aren't Any Playgrounds To Book
Welcome to GOFO Booking System
1- Login
2- Resigter
3- Exit
Please enter your choice :
```

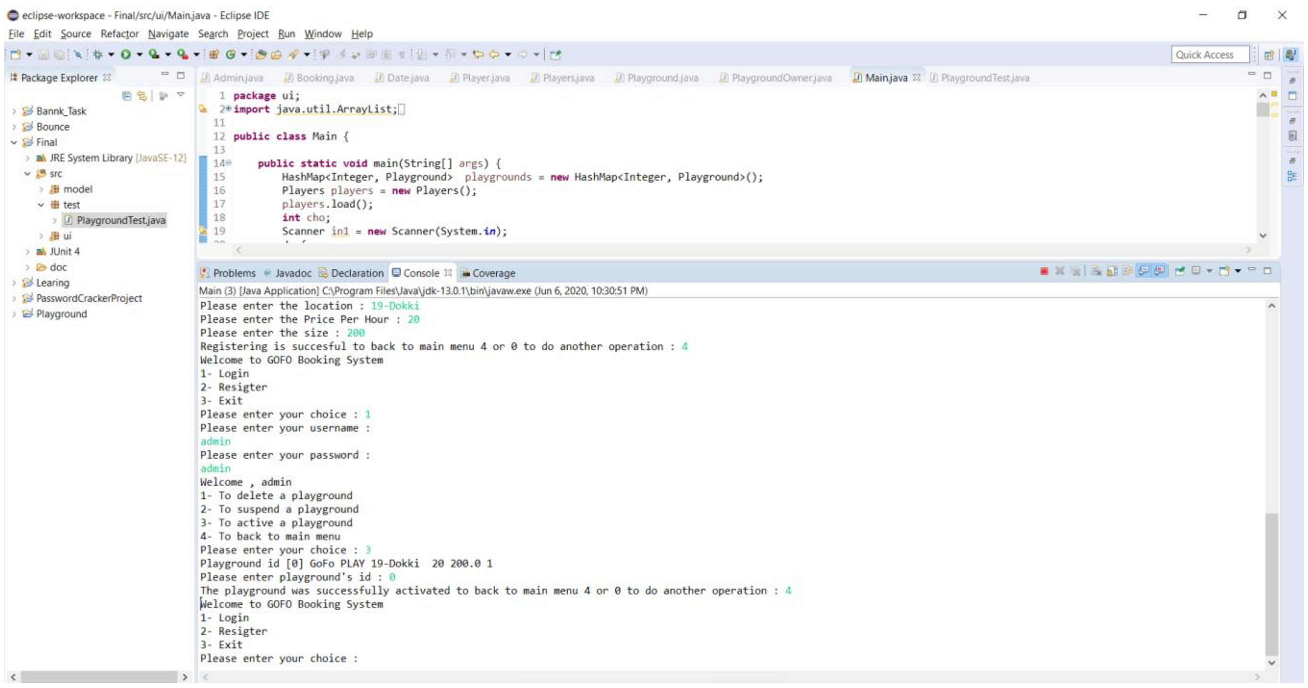


```
package ui;
import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        HashMap<Integer, Playground>  playgrounds = new HashMap<Integer, Playground>();
        Players players = new Players();
        players.load();
        int cho;
        Scanner in1 = new Scanner(System.in);
```

Main (3) [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Jun 6, 2020, 10:30:51 PM)

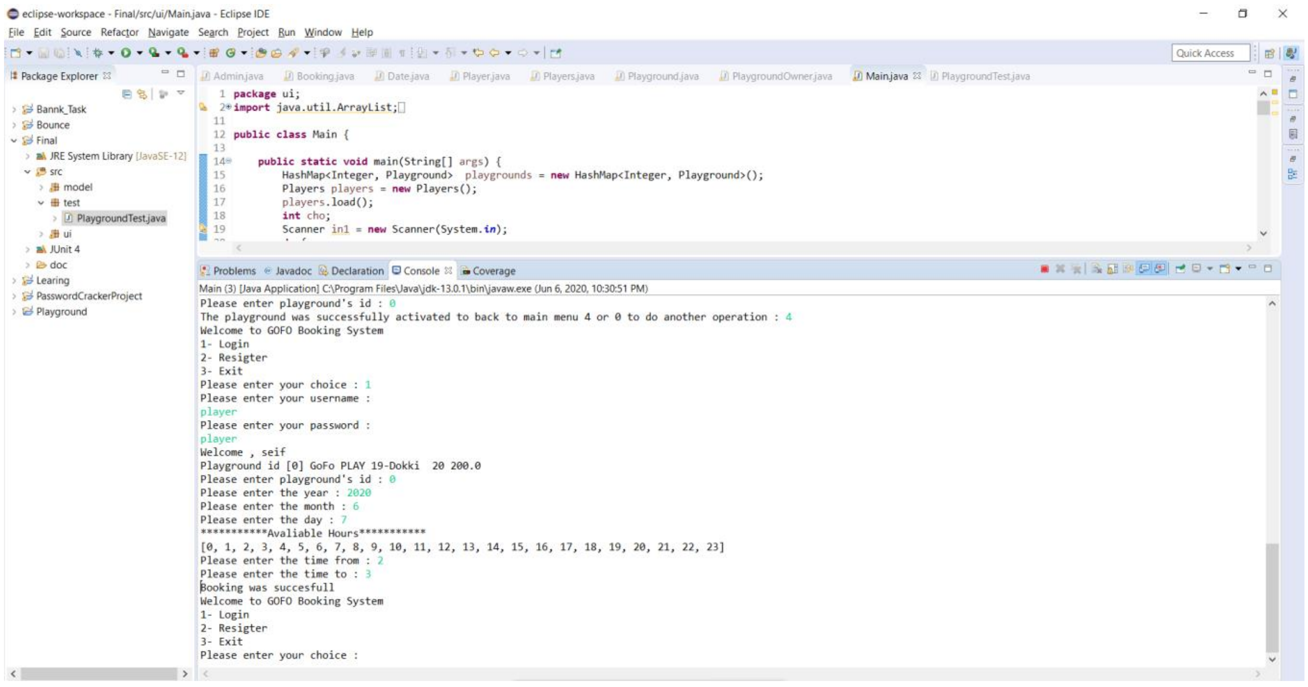
```
Please enter your password :
player
Welcome , seif
There Aren't Any Playgrounds To Book
Welcome to GOFO Booking System
1- Login
2- Resigter
3- Exit
Please enter your choice : 1
Please enter your username :
ahmed
Please enter your password :
ahmed
Welcome , ahmed
1- Resigt playground
2- back to main menu
Please enter your choice : 1
Please enter the name : GoFo PLAY
Please enter the location : 19-Dokki
Please enter the Price Per Hour : 20
Please enter the size : 200
Registering is succesful to back to main menu 4 or 0 to do another operation : 4
Welcome to GOFO Booking System
1- Login
2- Resigter
3- Exit
Please enter your choice :
```

Software Design Specification


eclipse-workspace - Final/src/ui/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
Bannk_Task
Bounce
Final
JRE System Library [JavaSE-12]
src
model
test
PlaygroundTest.java
ui
JUnit 4
doc
Learing
PasswordCrackerProject
Playground
Admin.java Booking.java Date.java Player.java Players.java Playground.java PlaygroundOwner.java Main.java PlaygroundTest.java
Problems Javadoc Declaration Console Coverage

```
package ui;
import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        HashMap<Integer, Playground>  playgrounds = new HashMap<Integer, Playground>();
        Players players = new Players();
        players.load();
        int cho;
        Scanner in1 = new Scanner(System.in);
```

```
Main (3) [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Jun 6, 2020, 10:30:51 PM)
Please enter the location : 19-Dokki
Please enter the Price Per Hour : 20
Please enter the size : 200
Registering is succesful to back to main menu 4 or 0 to do another operation : 4
Welcome to GOFO Booking System
1- Login
2- Resigter
3- Exit
Please enter your choice : 1
Please enter your username :
admin
Please enter your password :
admin
Welcome , admin
1- To delete a playground
2- To suspend a playground
3- To active a playground
4- To back to main menu
Please enter your choice : 3
Playground id [0] GoFo PLAY 19-Dokki  20 200.0 1
Please enter playground's id : 0
The playground was successfully activated to back to main menu 4 or 0 to do another operation : 4
Welcome to GOFO Booking System
1- Login
2- Resigter
3- Exit
Please enter your choice :
```


eclipse-workspace - Final/src/ui/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
Bannk_Task
Bounce
Final
JRE System Library [JavaSE-12]
src
model
test
PlaygroundTest.java
ui
JUnit 4
doc
Learing
PasswordCrackerProject
Playground
Admin.java Booking.java Date.java Player.java Players.java Playground.java PlaygroundOwner.java Main.java PlaygroundTest.java
Problems Javadoc Declaration Console Coverage

```
package ui;
import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        HashMap<Integer, Playground>  playgrounds = new HashMap<Integer, Playground>();
        Players players = new Players();
        players.load();
        int cho;
        Scanner in1 = new Scanner(System.in);
```

```
Main (3) [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Jun 6, 2020, 10:30:51 PM)
Please enter playground's id : 0
The playground was successfully activated to back to main menu 4 or 0 to do another operation : 4
Welcome to GOFO Booking System
1- Login
2- Resigter
3- Exit
Please enter your choice : 1
Please enter your username :
player
Please enter your password :
player
Welcome , seif
Playground id [0] GoFo PLAY 19-Dokki  20 200.0
Please enter playground's id : 0
Please enter the year : 2020
Please enter the month : 6
Please enter the day : 7
***********Avaliable Hours***********
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
Please enter the time from : 2
Please enter the time to : 3
Booking was succesfull
Welcome to GOFO Booking System
1- Login
2- Resigter
3- Exit
Please enter your choice :
```

CS251: Phase 2 – <Tankers>
Project: <GoFo>

Software Design Specification

Authors

- Seif Mosaad, Ahmed Nabil, Habiba Amr, Marina Moheb.