Cairo University
Faculty of Computers and Informatic

PM-209

CS251

Software Engineering I

Gofo

Software Design Specifications

Version 2.0

June 2020

NAME	EMAIL
Neimat Soliman	neimatsoliman8040@gmail.com
Alaa Reda	Okcat99@gmail.com
Eman Salah	Emisalah2000@gmail.com
Noura Saad	nourasa3d19@gmail.com

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

Software Design Specification

Contents

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

Software Design Specification

Team ID	Name	Email	Mobile
20180315	Neimat	neimatsoliman8040@gmail.com	01156945877
20170449	Alaa	Okcat99@gmail.com	01122257525
20180063	Eman	Emisalah2000@gmail.com	01110540927
20180317	Noura	nourasa3d19@gmail.com	01111025924

Document Purpose and Audience

• This document is about system that is a booking system for football playgrounds, anyone can register himself on the system and create a profile, he can see the playgrounds near to

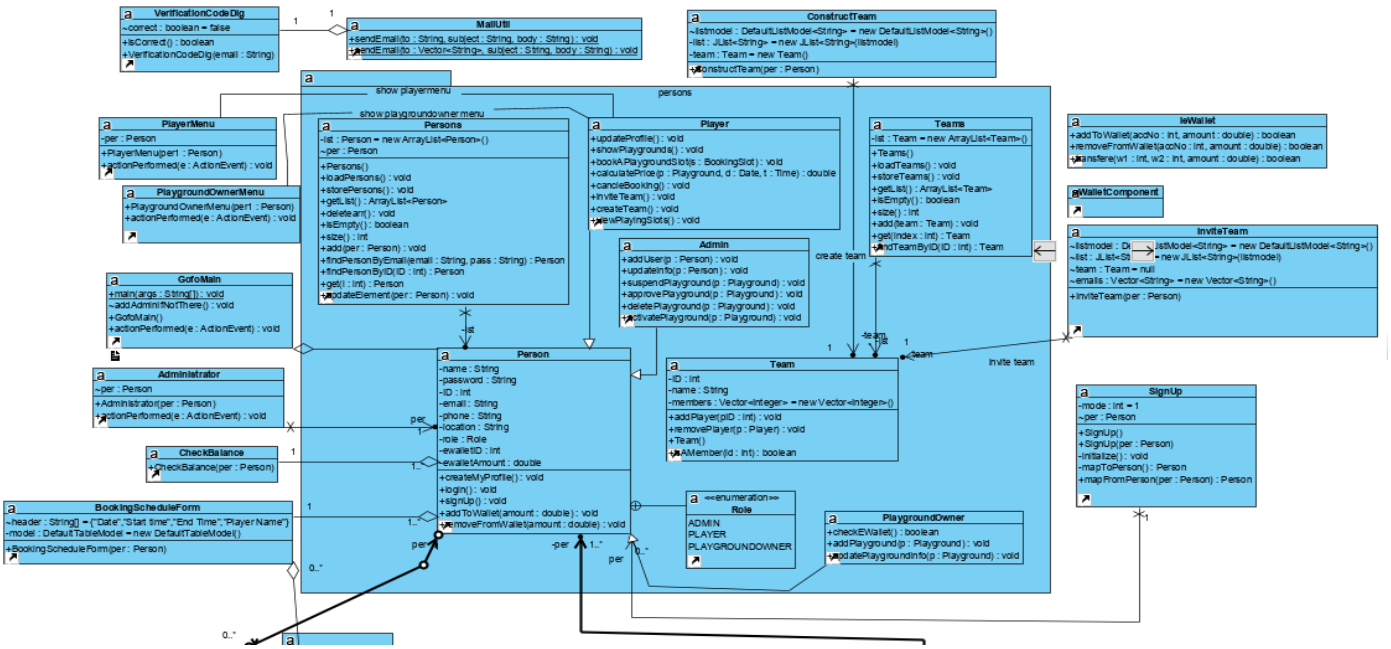
• him or in a specific area or all of them.

• Audience that will read this document they will be players and playground owners.

System Models

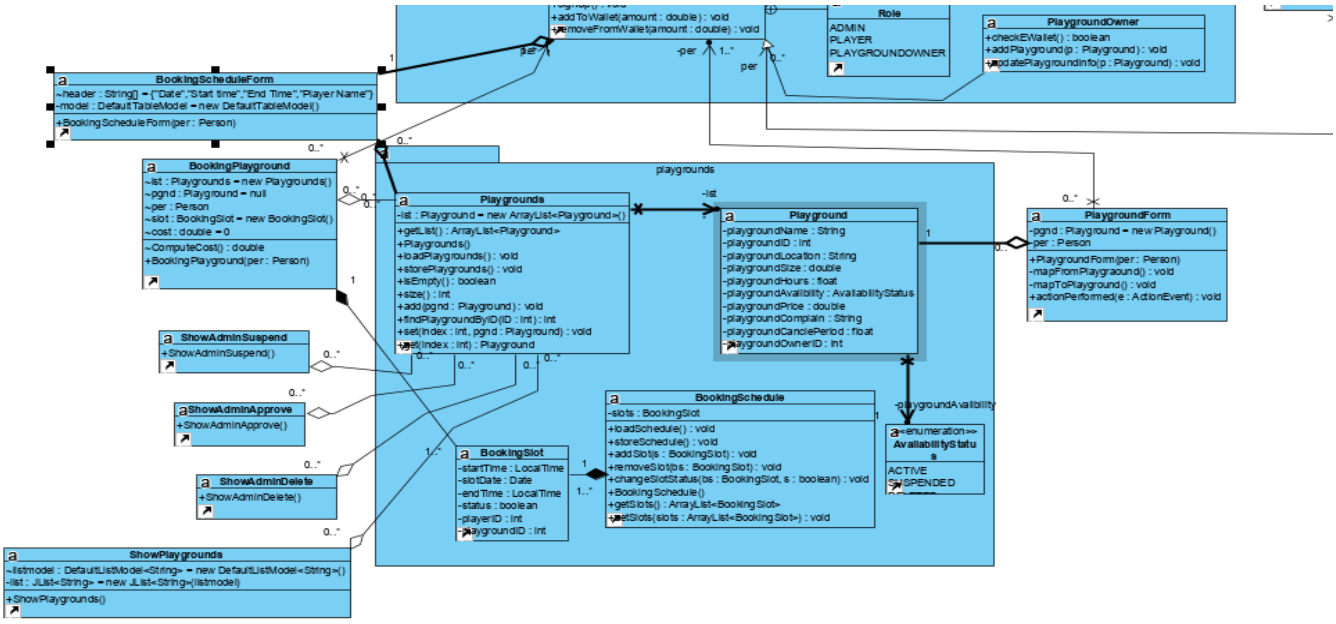
I. Class Diagram(s)

Picture1



Software Design Specification

Picture 2



II. Class Descriptions

Class ID	Class Name	Description & Responsibility
1.CID#1	GogoMain	1.addAdminIfNotThere():for add the admin in the gofo if not there any admin 2.GofoMain():constructor for creating a frame to show for the user 3.actionPerformed(): the action listener if the user put in sign in open for the user (player,playgroundowner,admin), if the user put in sign up add the new user

Software Design Specification

Class ID	Class Name	Description & Responsibility
2.CID#2	Administrator	1. Administrator(): constructor for creating the frame for admin options 2. Actionperformed(): the action listener if the admin put in update info the admin can update his info If the admin put in approve show the playgrounds for him the playground that is suspended If the admin put in delete show the playgrounds for him the playground that is suspended and active If the admin put in suspend show the playgrounds for him the playground that is active
3.CID#3	BookingPlayground	1. ComputeCost(): calculate the price for booking the playground 2. BookingPlayground():constructor creating the frame for showing for the player booing playground to make a player book the special playground
4.CID#4	BookingScheduleForm	1. BookingScheduleForm():constructor for creating the frame for showing for the player the booking schedule
5.CID#5	CheckBalance	1. CheckBalance():constructor for creating the frame to make the person to check his ewallet
6.CID#6	ConstructTeam	1. ConstructTeam():constructor for creating the frame for show the player to create the team
7.CID#7	eWalletComponent	
8.CID#8	IeWallet	
9.CID#9	InviteTeam	1.InviteTeam():constructor for creating the frame to can the player to invite team
10.CID#10	MailUtil	1. sendEmail(String to,String Subject ,String Body) :for make the player send the email to invite and create the team 2. sendEmail(vector<string>to,string Subject,String Body): for make the player send the email for more than one player
11.CID#11	PlayerMenu	1. PlayerMenu():constructor for creating the frame for showing the playeroptions
12.CID#12	PlaygroundForm	1.PlaygroundForm(): constructor for adding the playground 2.mapToPlaygraound():function to take the information of playground that taken from the playground owner 3.mapFromPlaygraound(): to select the availability status from playground 4.actionPerformed():the action listener if the playground owner put in add the program add the playground In the list

Software Design Specification

Class ID	Class Name	Description & Responsibility
13.CID#13	PlaygroundOwnerMenu	1.PlaygroundOwnerMenu():constructor for creating the frame for showing the playgroundowner options
14.CID#14	ShowAdminApprove	1. ShowAdminApprove():constructor for creating the frame for show for the admin the playground that it is suspended
15.CID#15	ShowAdminDelete	1. ShowAdminDelete():constructor for creating the frame for show for the admin the playground that it is suspended or active
16.CID#16	ShowAdminSuspend	1. ShowAdminSuspend():constructor for creating the frame for show for the admin the playground that it is active
17.CID#17	ShowPlaygrounds	1. ShowPlaygrounds():constructor for creating the frame that show the playgrounds for the player
18.CID#18	SignUp	1. SignUp(): Constructor for the new user enter in the gofo 2. SignUp(Person): Parametriz constructor for updating the user information 3. Initialize(): To diffrenece between the new user and the user want to update his information 4. mapToPerson(): for the new user that entered his information to add to list 5. mapFromPerson(): for bring the information for special user to update his information
19.CID#19	VerificationCodeIDg	1. isCorrect():to check if the VerificationCode that send to email is equal to the input VerificationCode from the user 2. VerificationCodeIDg():constructor for creating the frame for enter the uer the VerificationCode that send to him
20.CID#20	Admin	
21.CID#21	Person	1. createMyProfile(): 2. login(): 3. signUp(): 4. addToWallet():to add the mony for the user 5. removeFromWallet():to remove the money for the user

Software Design Specification

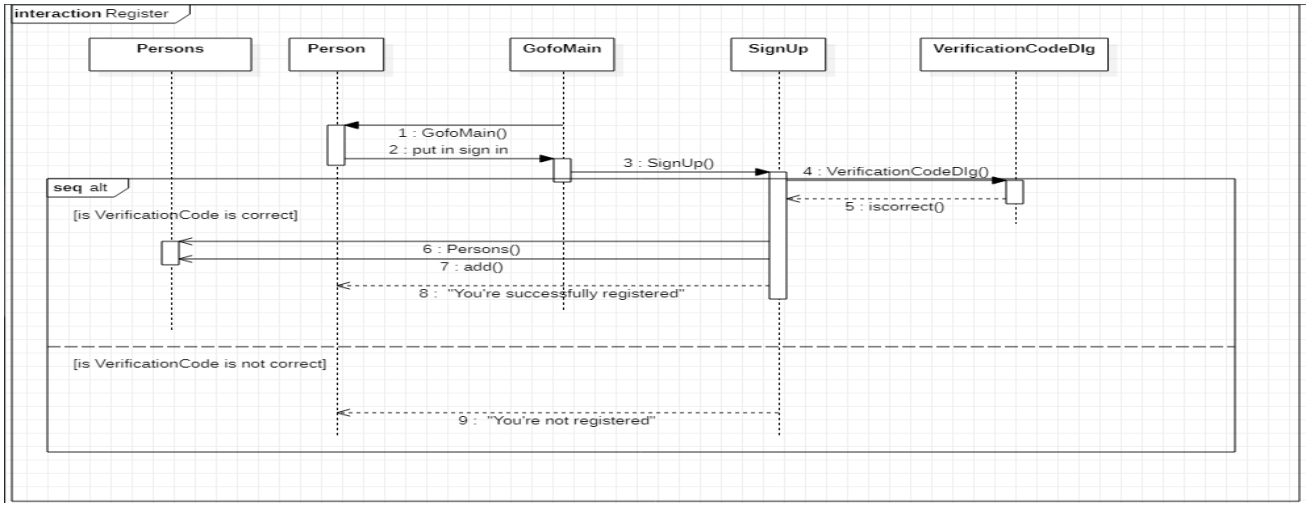
Class ID	Class Name	Description & Responsibility
22.CID#22	Persons	1. Persons():constructor for the function load person 2. loadPersons():for reading from file 3. storePersons():for write in file and save 4. getList():to return list of persons 5. deletearr():to delete all elements of arraylist 6. isEmpty():to check if the arraylist is empty or not 7. size() to return the size of arraylist 8. add():to add person to arraylist 9. findPersonByEmail():to find the person by email 10. findPersonByID():to find the person by id 11. get():return person by specific id 12. updateElement():update person by specific id
23.CID#23	Player	
24.CID#24	PlaygroundOwner	
25.CID#25	Team	1. addPlayer():to add player 2. removePlayer(): 3. Team():constructor 4. isAMember():check if the person a member of the team
26.CID#26	Teams	1. Teams():constructor for the function load Teams 2. loadTeams():for reading from file 3. storeTeams():for write in file and save 4. getList():to return list of Teams 5. isEmpty():to check if the arraylist is empty or not 6. size() to return the size of arraylist 7. add():to add Team to arraylist 8. get():return team by specific id 9. findPersonByID():to find the team by id
27.CID#27	AvailabilityStatus	
28.CID#28	BookingSchedule	1. loadSchedule():for reading from file 2. storeSchedule():for write in file and save 3. addSlot():for adding slot to arraylist for playground 4. removeSlot(): 5. changeSlotStatus(): 6. BookingSchedule():for function loadSchedule 7. getSlots():to return allslots of arraylist 8. setSlots():to set slot for playground
29.CID#29	BookingSlot	
30.CID#30	Playground	

Software Design Specification

Class ID	Class Name	Description & Responsibility
31.CID#31	Playgrounds	1. getList():for return the list of playgrounds 2. Playgrounds()for the function loadPlaygrounds 3. loadPlaygrounds():for read from file 4. storePlaygrounds():for write in file and save 5. isEmpty():to check if the arraylist is empty or not 6. size():to return the size of arraylist 7. add():to add playground 8. findPlaygroundByID():to find playground by id 9. set():set playground by index 10. get():to return playground by id

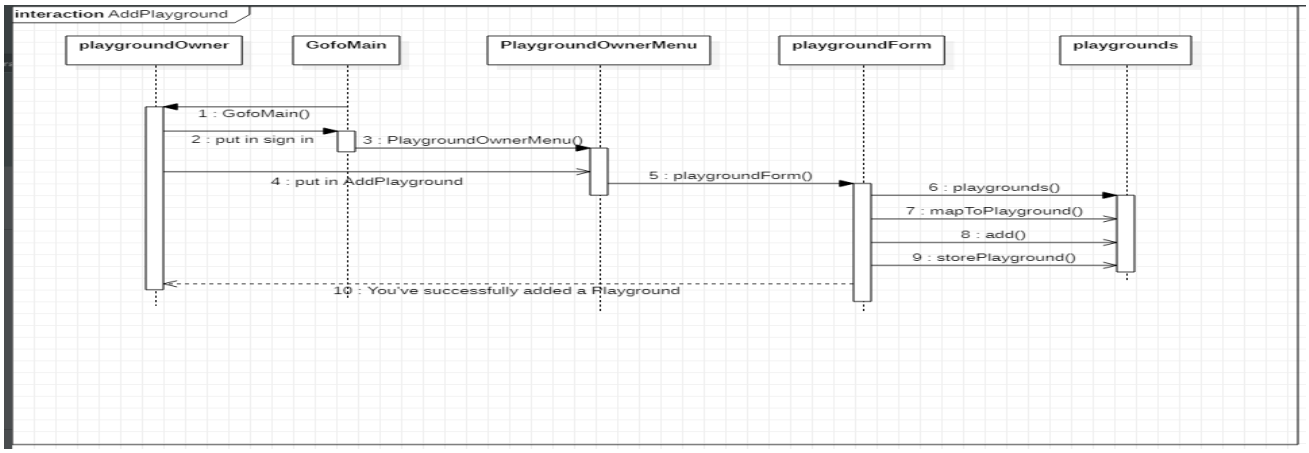
III. Sequence diagrams

1.register

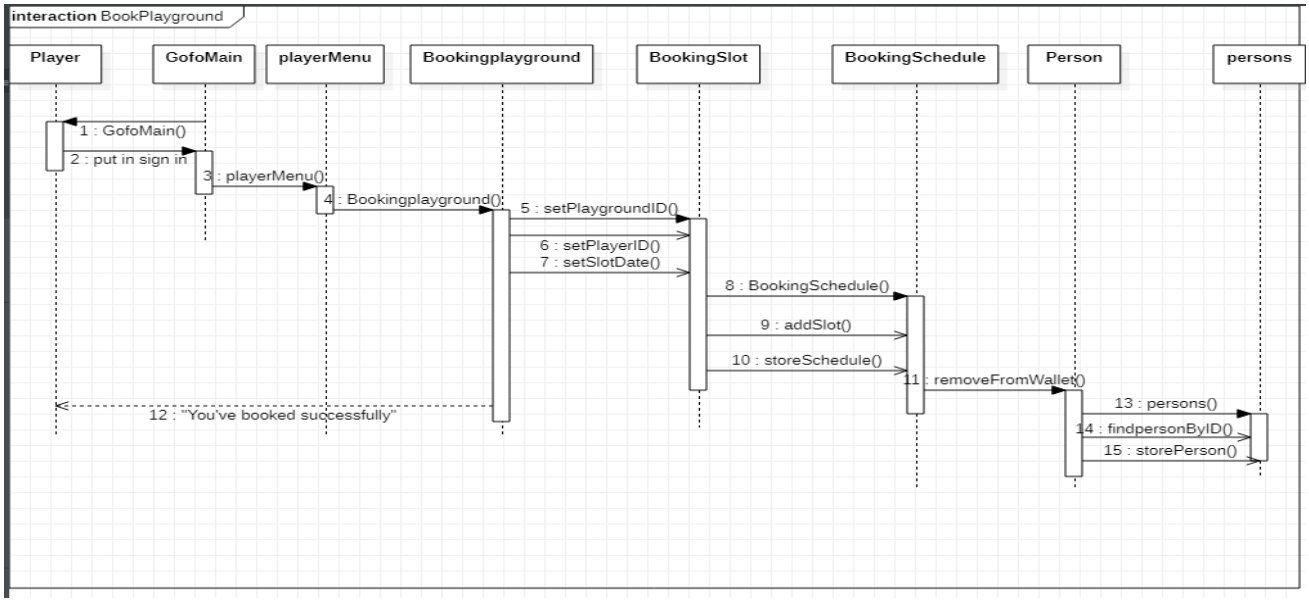


2.AddPlayground

Software Design Specification

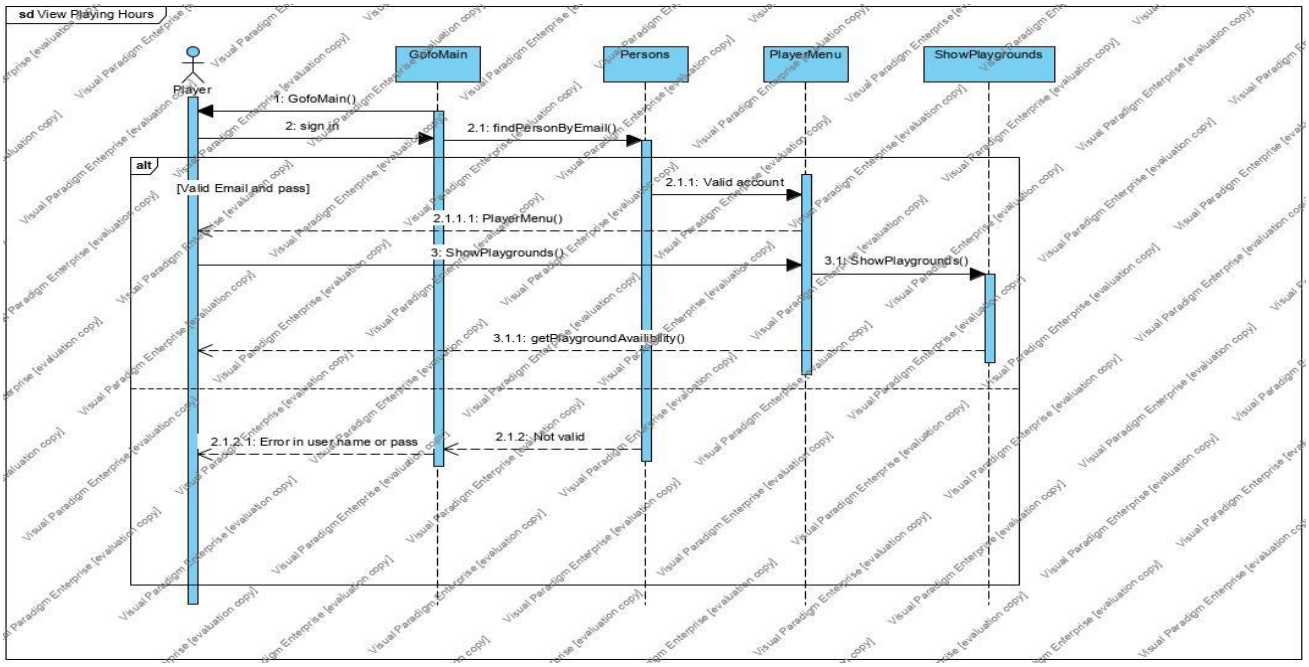


3.BookPlayground

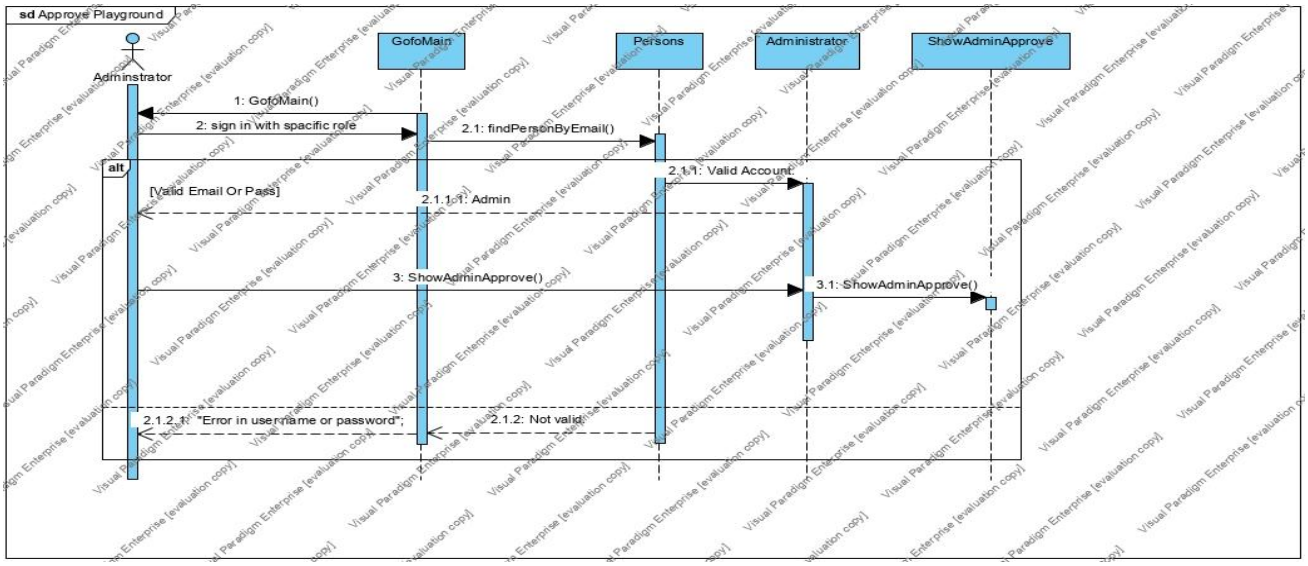


4.viewplayinghours

Software Design Specification

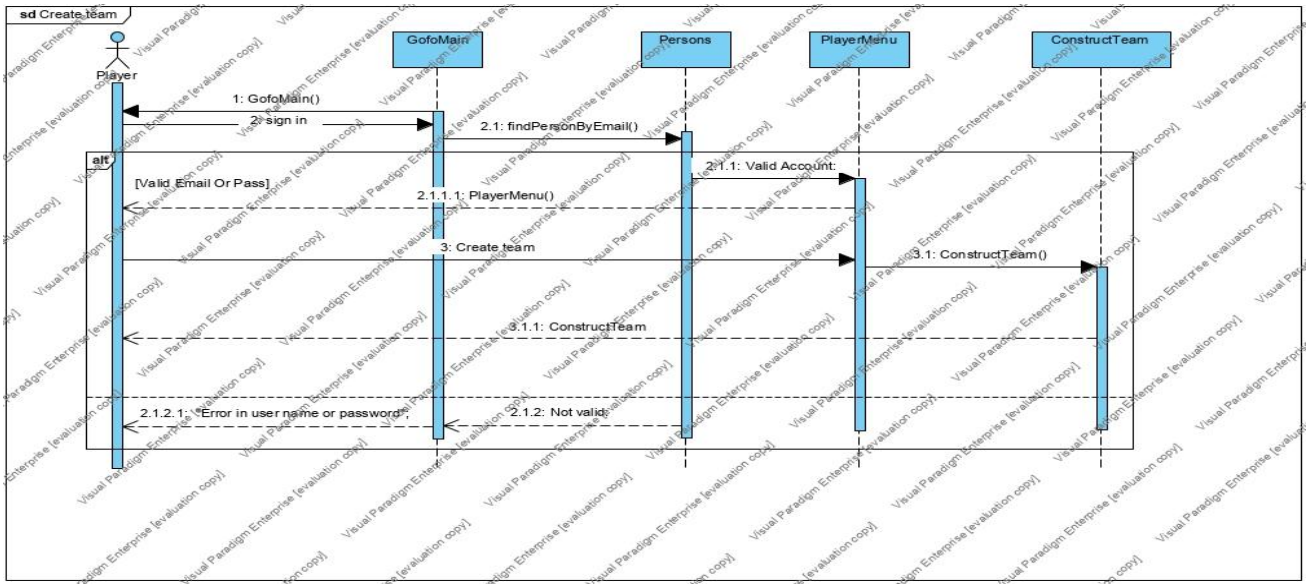


5.Approve Playground



Software Design Specification

6.Create team



Class - Sequence Usage Table

Sequence Diagram	Classes Used	All Methods Used
Register	GofoMain Person Persons SignUp VerificationCodeDlg	Constructor: GofoMain() Persons() SignUp() VerificationCodeDlg() Function: Iscorrect() add()

Software Design Specification

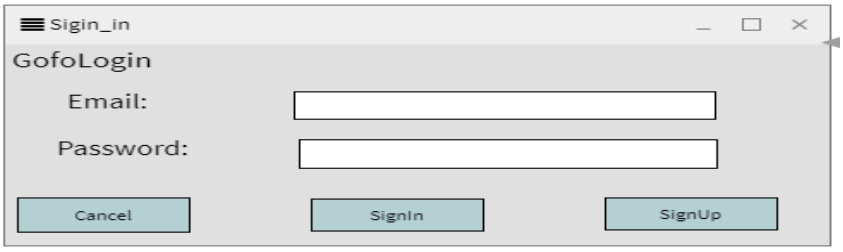
addPlayground	GofoMain PlaygroundOwner PlaygroundOwnerMenu PlaygroundForm Playgrounds	Constructor: GofoMain() PlaygroundOwnerMenu() PlaygroundForm() Playgrounds() Function: mapToplayground() add() storePlayground()
BookPlayground	GofoMain Player PlayerMenu Booking Playground BookingSlot BookingSchedule Person Persons	Constructor: GofoMain() PlayerMenu() Booking Playground() BookingSchedule() Persons() Function: setPlaygroundID() setPlayerID() setSlotDate() addSlot() storeSchedule() removeFromWallet() findpersonByID() storePerson()
viewplayinghours	GofoMain Player Persons PlayerMenu ShowPlaygrounds	Constructor: GofoMain() PlayerMenu() ShowPlaygrounds() Function: findPersonByEmail() getPlayground Availability()

Software Design Specification

Approve Playground	Adminstrator GofoMain Persons ShowAdminApprove	Constructor: GofoMain() ShowAdminApprove() Function: findPersonByEmail()
Create team	GofoMain Player Persons PlayerMenu ShowPlaygrounds ConstructTeam	Constructor: GofoMain() PlayerMenu() Function: findPersonByEmail()

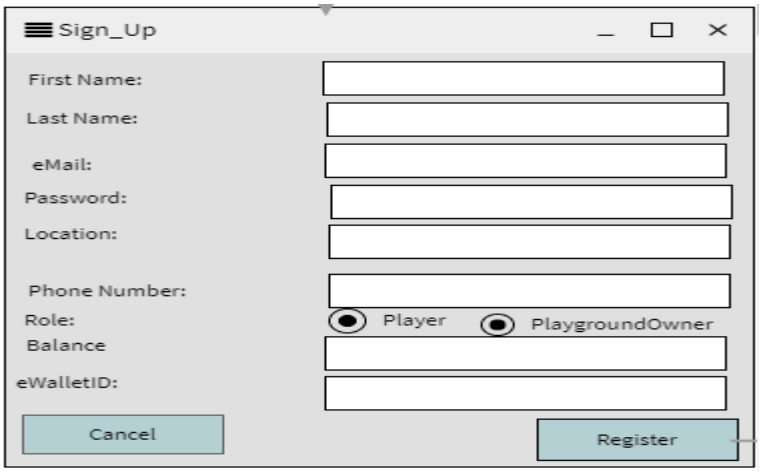
IV. User Interface Design

1.UI#1(sign_In)

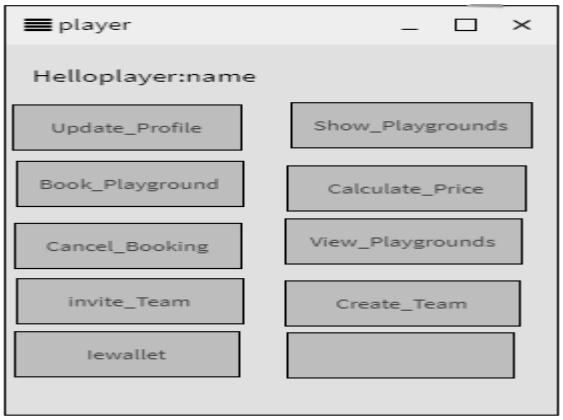


2.UI#2(Sign_up)

Software Design Specification



3.UI#3(PlayerOption)



4.UI#4(PlaygroundOwnerOption)

Software Design Specification

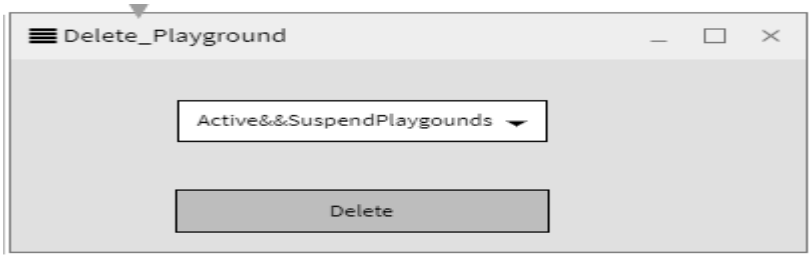
5.UI#5(AdminOption)

6.UI#6(AdminApprovePlayground)

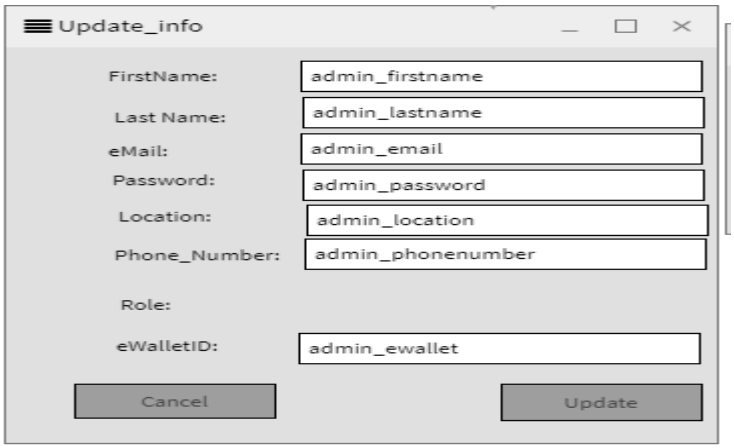
7.UI#7(AdminSuspendPlayground)

8.UI#8(AdmindeletePlayground)

Software Design Specification

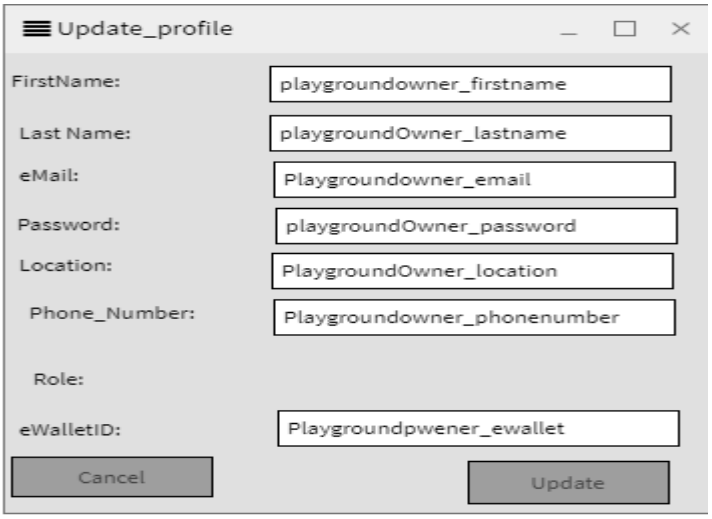


9.UI#9(AdminUpdateInfo)

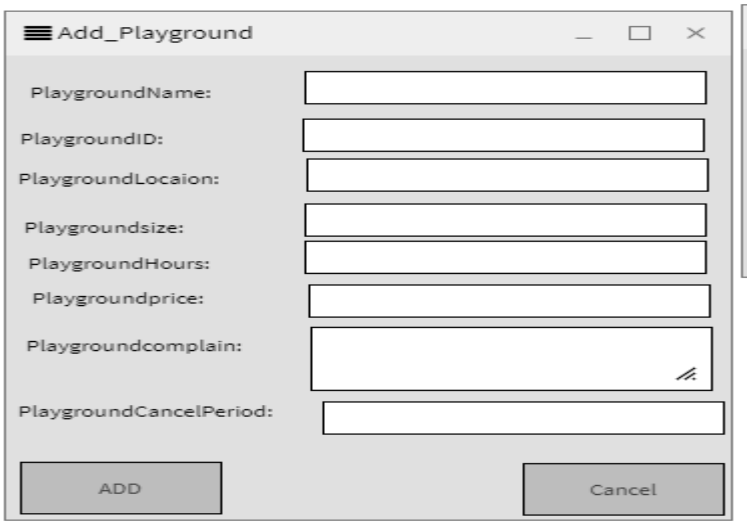


10.UI#10(PlaygroundOwnerUpdate)

Software Design Specification

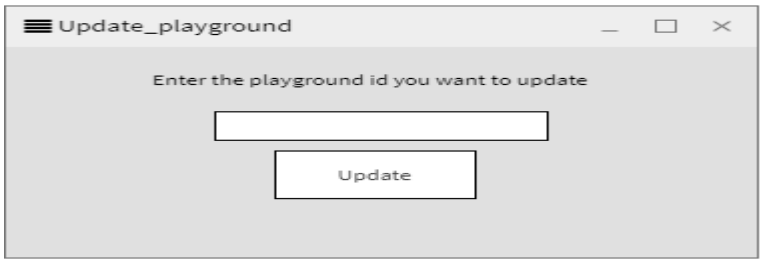


11.UI#11(PlaygroundOwnerADD)

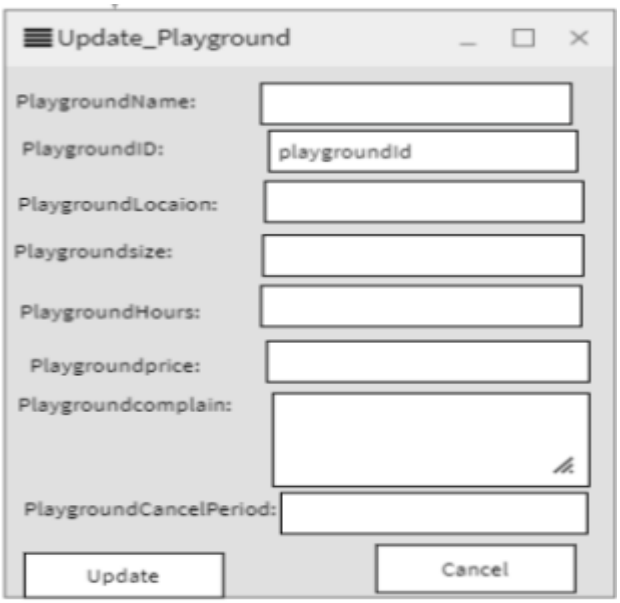


12.UI#12(playgroundOwnerUpdate1)

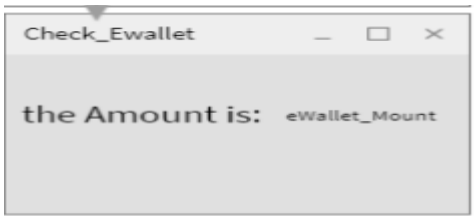
Software Design Specification



13.UI#13(playgroundOwnerUpdate2)

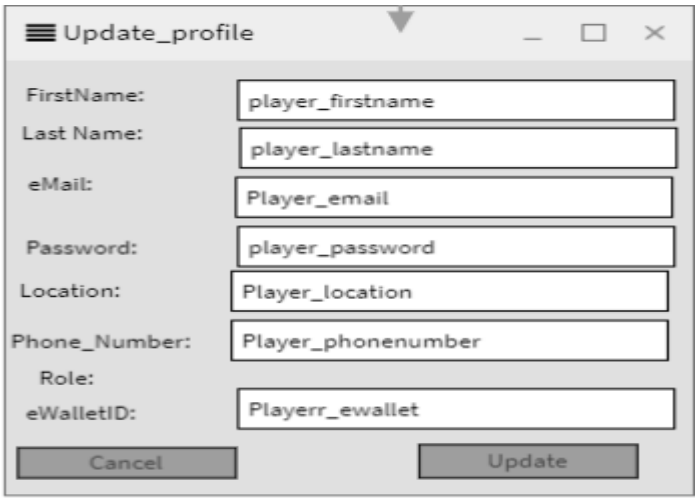


14.UI#14(PlaygroundOwnerCheck)

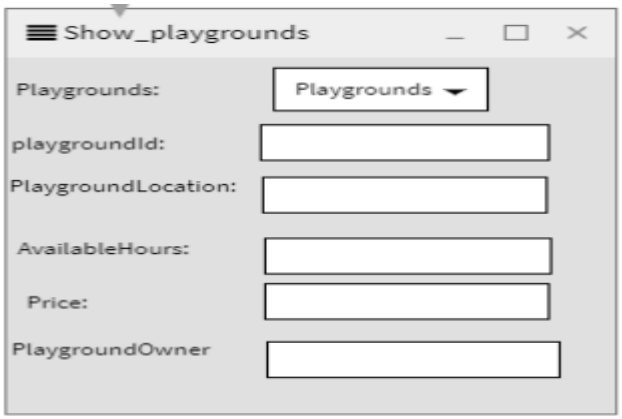


14.UI#14(PlayerUpdate)

Software Design Specification

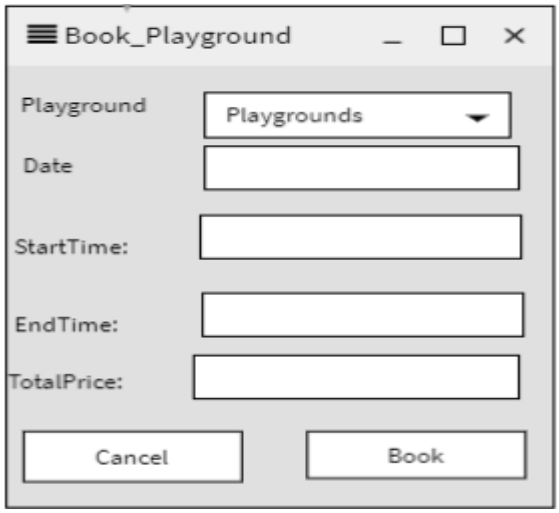


15.UI#15(PlayerShow)

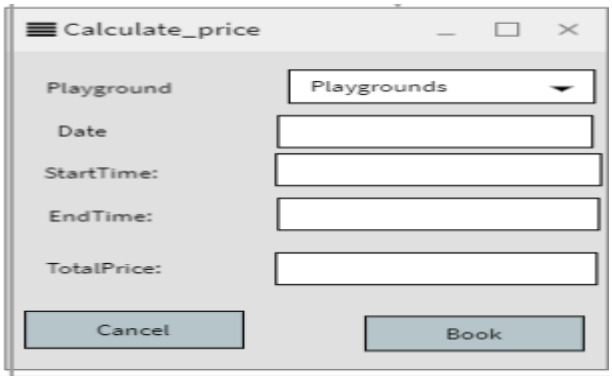


Software Design Specification

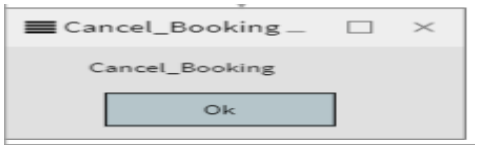
16.UI#16(PlayerBook)



17.UI#17(PlayerCalculate)

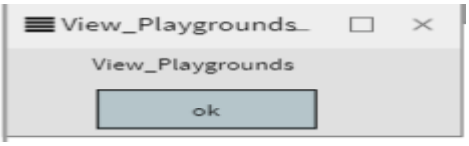


18.UI#18(PlayerCancel)



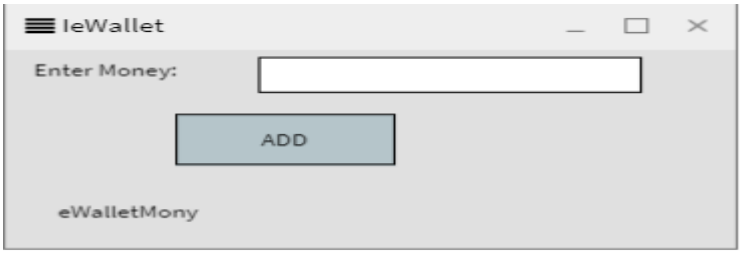
19.UI#19(PlayerView)

Software Design Specification

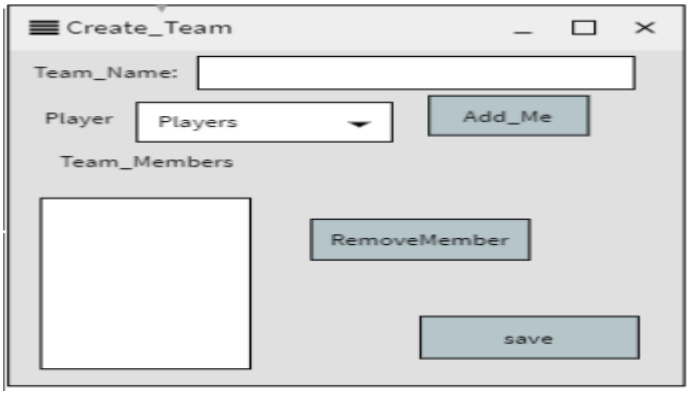


20.UI#20(PlayerInvite)

21.UI#21(PlayerIewallet)

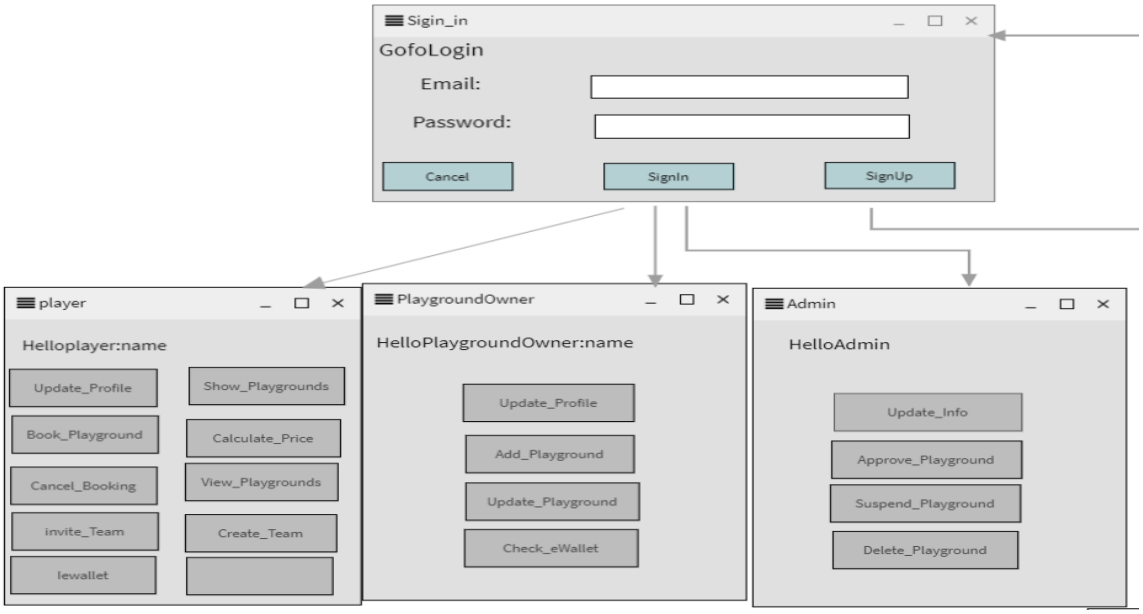


22.UI#22(PlayerCreate)

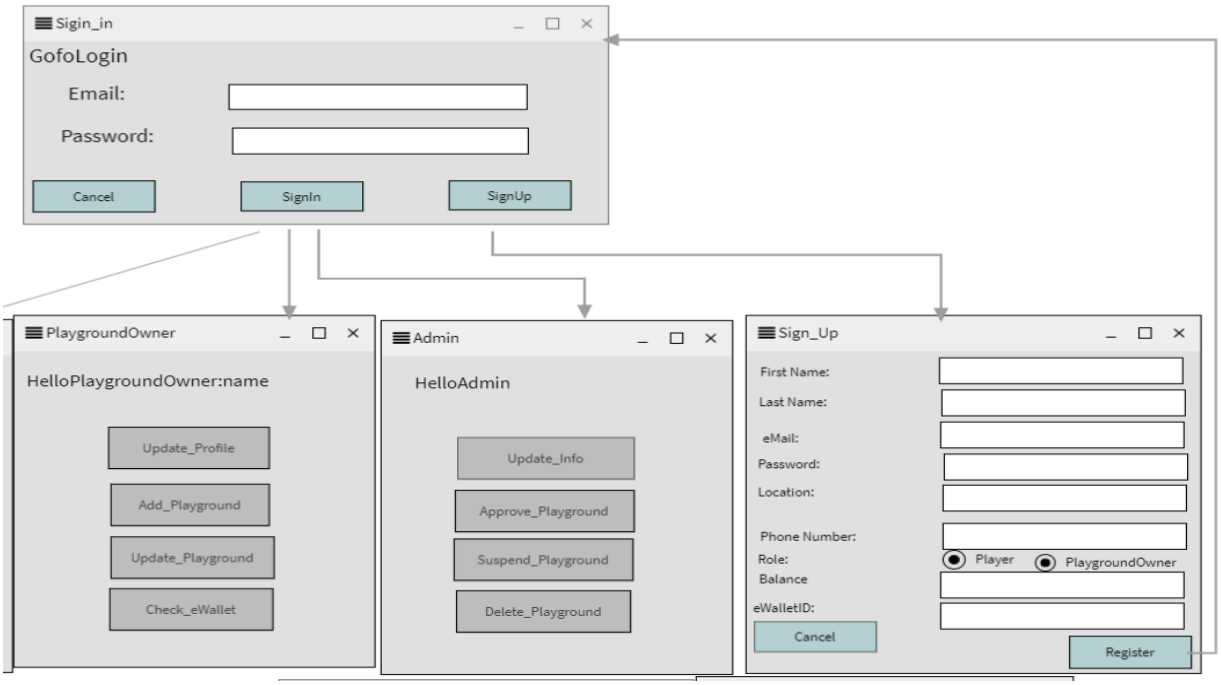


23.navMap#1(signin)

Software Design Specification

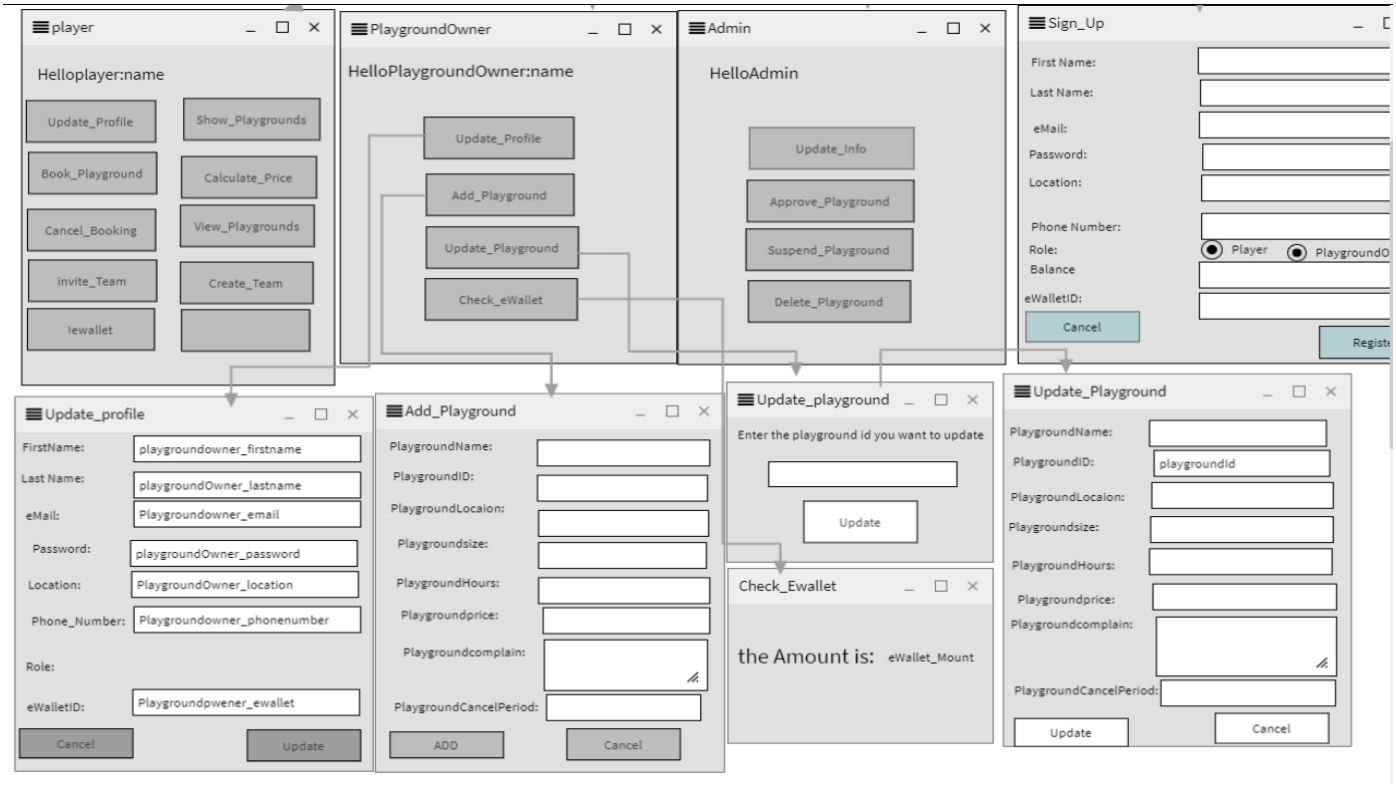


24.navMap#2(sign_up)



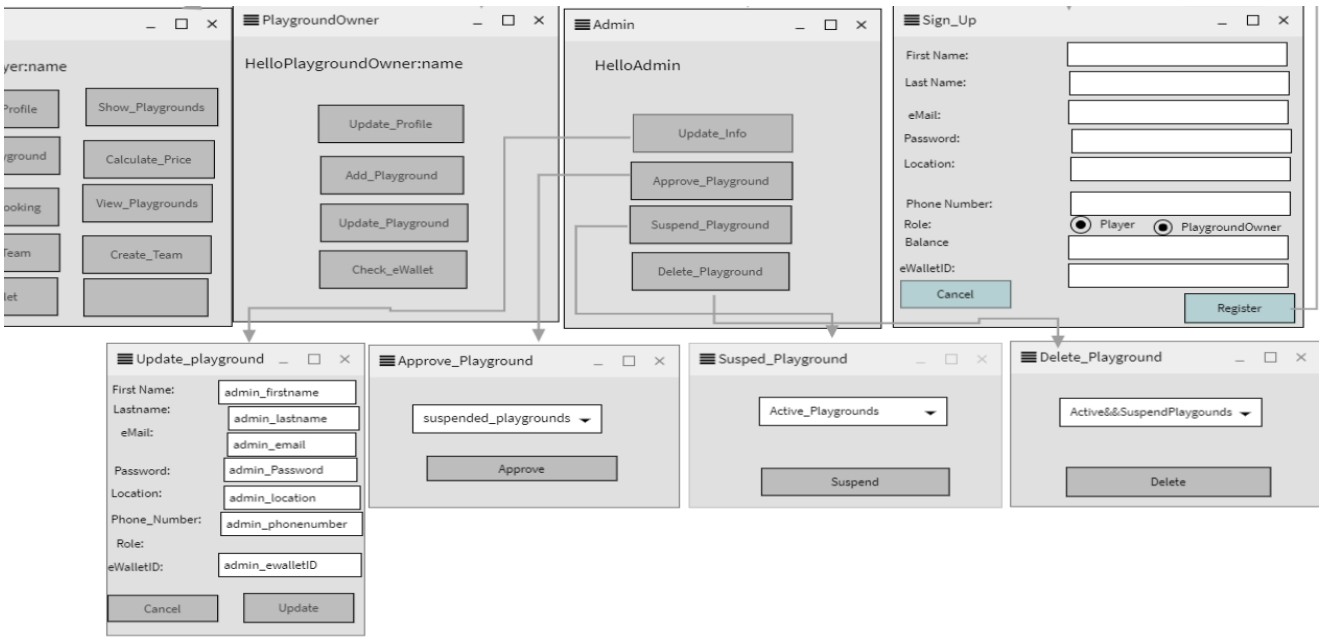
Software Design Specification

25.navMap#3(playgroundOption)



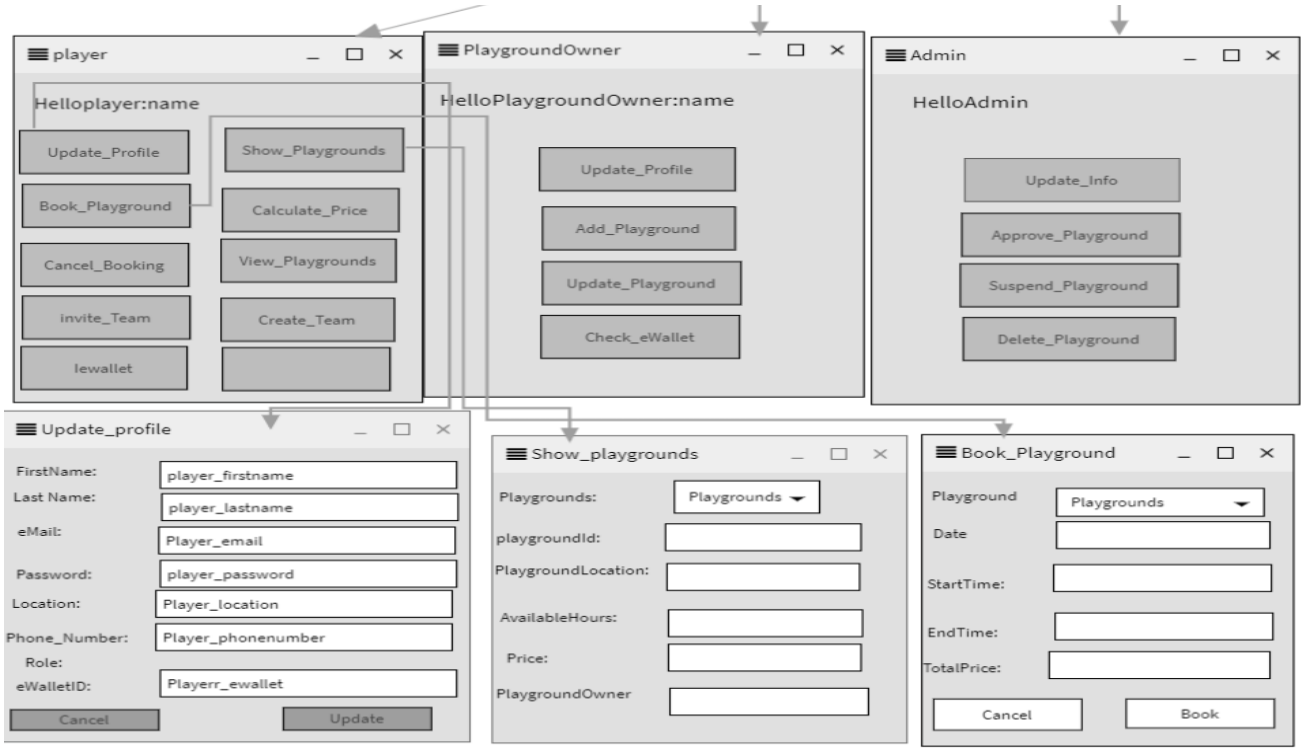
Software Design Specification

26.navMap#4(adminOption



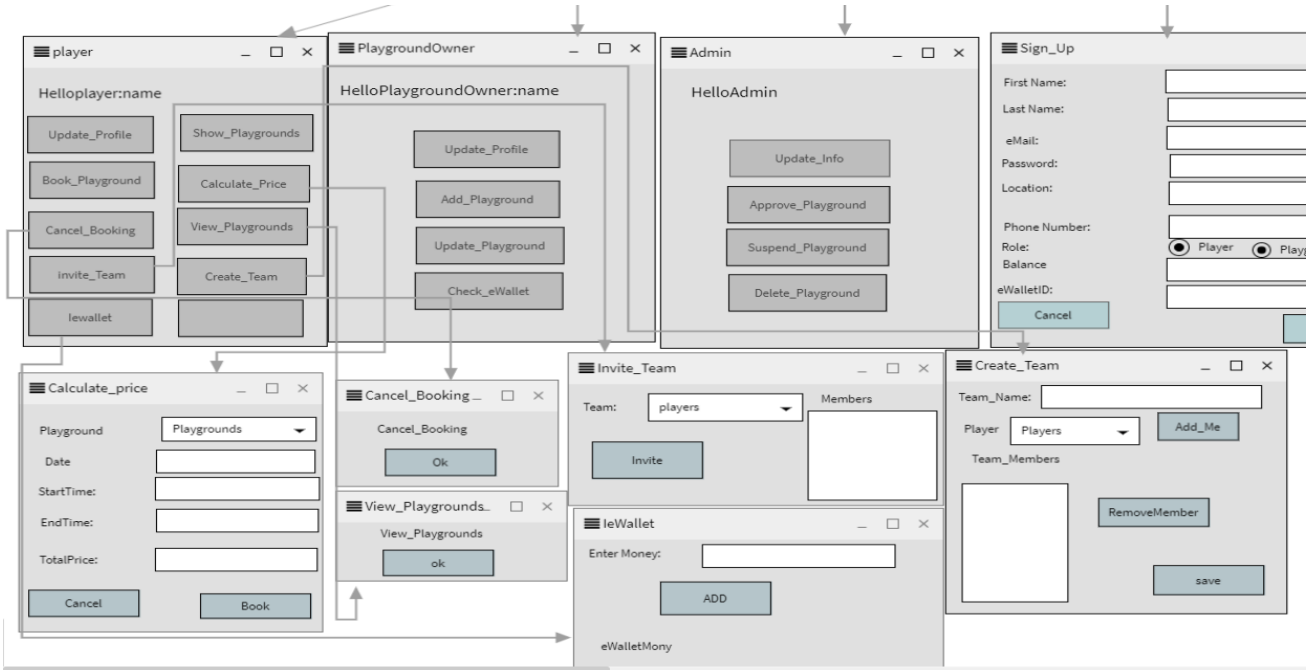
Software Design Specification

27.navMap#5(playerOption_1)



Software Design Specification

28.navMap#6(player_Option_2)



Screen ID	Screen Name	Screen / Wireframe Description
UI#1	sign_In	the user enter the email , the password and put sign in but if he new user he sign up first and sign in if he does not want to sign in or sign up he put cancel
UI#2	Sign_up	the new enter his information to register in the system

Software Design Specification

UI#4	PlaygroundOwnerOption	if the user sign up and choose the role as a playgroundowner and sign in the system open for you the options of playgroundowner
UI#5	AdminOption	There are one admin and he is unique so he sign in and the system open for you the admin option
UI#6	AdminApprovePlayground	One of the Admin options is the admin approve if the playground owner add a playground
UI#7	AdminSuspendPlayground	One of the Admin options is the admin suspend the playground if this playground booked from player
UI#8	AdmindeletePlayground	One of the Admin options is the admin delete playground if the admin find a lot of complains of this playground
UI#9	AdminUpdateInfo	One of the Admin options is the admin update his information
UI#10	PlaygroundOwnerUpdate	one of the playgroundowner options is the playground owner update his information
UI#11	PlaygroundOwnerADD	one of the playgroundowner options is the playground owner add a new playground

Software Design Specification

UI#12	playgroundOwnerUpdate1	one of the playgroundowner options is the playground owner update playground by two levels level one the playgroundOwner enter the id of the playground which want to update
UI#13	playgroundOwnerUpdate2	Level 2 he enter the data he want to change
UI#14	PlaygroundOwnerCheck	one of the playgroundowner options is the playground owner check the Ewallet Money
UI#15	PlayerUpdate	one of the player options is the player update his information
UI#16	PlayerShow	one of the player options is the system show Available playgrounds for the user
UI#17	PlayerBook	one of the player options is the player book a playground
UI#18	PlayerCalculate	one of the player options is the player calculate the price for a playground
UI#19	PlayerCancel	one of the player options is the player cancel the booking for playground
UI#20	PlayerView	one of the player options is the system view Playgrounds for the player

Software Design Specification

UI#21	PlayerInvite	one of the player options is the the player can invite team for playing
UI#22	PlayerIewallet	one of the player options is the the player can add money to his ewallet
UI#23	PlayerCreate	one of the player options is the the player can create a team

Tools

- starUml
- Visual-Paradigm
- Wireframe
- Eclipse
- netbeans

Ownership Report

Item	Owners
Neimat soliman	Sequence diagram 3 Code 4 classes Class diagram github
Alaa reda	Code 14 classes
Eman salah	Sequence 3 Class diagram User interface Navigation map Code 9 classes

Software Design Specification

Nora saad	Code 4 classes Documentation Screens for output and video

Appendix A: Code Listing and Screen Snapshots

Class: Administrator

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import persons.Person;
import javax.swing.JLabel;
import javax.swing.JButton;
/**
 * this is class Administrator class
 * Date:10-june-2020
 * @author Eman,noura, alaa,neimat
 * @version 1.0
 *
 *
 */

@SuppressWarnings("serial")
public class Administrator extends JFrame implements ActionListener {
    private JPanel contentPane;
    /**
     * label named Hello_admin
     */
    private JLabel label=new JLabel("Hello_admin");
    /**
     * object from class person
     */
    Person per;

    /**
     * Launch the application.
     */
```

Software Design Specification

```
    /**
     * Create the frame for admin page
     * @param per is an object from person class to set admin details
     */
    public Administrator(Person per) {
        this.per=per;
        setTitle("Admin");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 420, 300);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        label.setBounds(50,10, 131, 30);
        contentPane.add(label);
        JButton btnNewButton = new JButton("Update Info");
        btnNewButton.setBounds(146, 48, 131, 30);
        contentPane.add(btnNewButton);
        btnNewButton.addActionListener(this);
        JButton btnNewButton_1 = new JButton("Approve Playground");
        btnNewButton_1.setBounds(146, 88, 131, 30);
        contentPane.add(btnNewButton_1);
        btnNewButton_1.addActionListener(this);
        JButton btnNewButton_2 = new JButton("Suspend Playground");
        btnNewButton_2.setBounds(146, 128, 131, 30);
        contentPane.add(btnNewButton_2);
        btnNewButton_2.addActionListener(this);
        JButton btnNewButton_3 = new JButton("Delete Playground");
        btnNewButton_3.setBounds(146, 168, 131, 30);
        contentPane.add(btnNewButton_3);
        btnNewButton_3.addActionListener(this);
    }


    @Override
    /**
     * to action listener with the buttons
     */
    public void actionPerformed(ActionEvent e) {
```

Software Design Specification

```
        // TODO Auto-generated method stub
        if(e.getActionCommand()=="Update Info")
        {
            new SignUp(per);
        }
        if(e.getActionCommand()=="Approve Playground")
        {

             new ShowAdminApprove();
        }
        if(e.getActionCommand()=="Suspend Playground")
        {
             new ShowAdminSuspend();
        }
        if(e.getActionCommand()=="Delete Playground")
        {

                    new ShowAdminDelete();
        }

    }

}
```

Class: BookingPlayground

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;

import persons.Person;
import persons.Persons;
import playgrounds.AvailabilityStatus;
import playgrounds.BookingSchedule;
import playgrounds.BookingSlot;
```

Software Design Specification

```
import playgrounds.Playground;
import playgrounds.Playgrounds;

import javax.swing.JLabel;
import javax.swing.JOptionPane;

import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.text.SimpleDateFormat;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.Vector;

import javax.swing.JComboBox;
import javax.swing.JTextField;
import javax.swing.JButton;
/**
 * this is a gui for  BookingPlayground class
 * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */
@SuppressWarnings("serial")
public class BookingPlayground extends JFrame {
/**
 * JPanel to display all components on a frame
 */
    private JPanel contentPane;
    /**
     * combobox to select from it
     */
    JComboBox<String> comboBox;
    /**
     * JTextField to enter text
     */
```

Software Design Specification

```
    private JTextField textField;
    private JTextField textField_1;
    private JTextField textField_2;
    private JTextField textField_3;
    /**
     * 1st is object from Playgrounds class
     */
    Playgrounds lst=new Playgrounds();
    /**
     * pgnd is an object from Playground class and set it to null
     */
    Playground pgnd=null;
/**
 * per is an object from Person class
 */
    Person per;
    /**
     * slot is an object from BookingSlot class
     */
    BookingSlot slot=new BookingSlot();
    /**
     * Double attribute (cost) and it set to zero
     */
    double cost=0;
    /**
     * function to Compute Cost of booking the playground
     *
     * @return returning double value (cost of booking playground )
     */
    double ComputeCost()
    {
        if(pgnd==null)pgnd=lst.get(comboBox.getSelectedIndex());
        try {
            slot.setStartTime(LocalTime.parse(textField_1.getText(), Date
TimeFormatter.ISO_LOCAL_TIME));
            slot.setEndTime(LocalTime.parse(textField_2.getText(), DateTi
meFormatter.ISO_LOCAL_TIME));
            double z=(double)(slot.getEndTime().toSecondOfDay()-
slot.getStartTime().toSecondOfDay())/3600;
```

Software Design Specification

```
            double cost=z*pgnd.getPlaygroundPrice();
            textField_3.setText(String.valueOf(cost));
            return cost;

        }catch(Exception e)
        {
            return 0;
        }

    }

    /**
     * this function of  Booking Playground and check if its Availability
Status
     * @param per is an object from person class
     */
    public BookingPlayground(Person per) {
        this.per=per;
        setTitle("Booking Playground");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 345, 345);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        comboBox= new JComboBox<String>();
        JLabel lblNewLabel = new JLabel("Playground");
        lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblNewLabel.setBounds(20, 27, 78, 19);
        contentPane.add(lblNewLabel);
        comboBox = new JComboBox<String>();
        Vector<Integer> indeces=new Vector<Integer>();
        for(int i=0;i<lst.size();i++)
        {
            Playground pg=lst.get(i);
            if(pg.getPlaygroundAvailbility()==AvailabilityStatus.ACTIVE)
            {
                comboBox.addItem(pg.getPlaygroundName());
                indeces.addElement(i);
```

Software Design Specification

```
            }
        }

        comboBox.setFont(new Font("Tahoma", Font.PLAIN, 13));
        comboBox.setToolTipText("");
        comboBox.setBounds(144, 28, 114, 20);
        comboBox.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent ev) {
                pgnd=lst.get(indeces.get(comboBox.getSelectedIndex()));
            }
        });
        contentPane.add(comboBox);
        JLabel lblNewLabel_1 = new JLabel("Date");
        lblNewLabel_1.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblNewLabel_1.setToolTipText("");
        lblNewLabel_1.setBounds(20, 74, 46, 14);
        contentPane.add(lblNewLabel_1);

        textField = new JTextField();
        textField.setBounds(144, 73, 114, 20);
        contentPane.add(textField);
        textField.setColumns(10);

        JLabel lblNewLabel_2 = new JLabel("Start Time");
        lblNewLabel_2.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblNewLabel_2.setBounds(20, 118, 69, 14);
        contentPane.add(lblNewLabel_2);

        textField_1 = new JTextField();
        textField_1.setBounds(144, 117, 114, 20);
        textField_1.getDocument().addDocumentListener(new DocumentListene
r(){
              @Override

               public void insertUpdate(DocumentEvent e) {
                  cost=ComputeCost();
               }

               @Override
```

Software Design Specification

```
                public void removeUpdate(DocumentEvent e) {
                        cost=ComputeCost();
                }

                @Override
                public void changedUpdate(DocumentEvent e) {
                        cost=ComputeCost();
                }
        });
        contentPane.add(textField_1);
        textField_1.setColumns(10);

        JLabel lblNewLabel_3 = new JLabel("End Time");
        lblNewLabel_3.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblNewLabel_3.setBounds(20, 165, 69, 14);
        contentPane.add(lblNewLabel_3);
        textField_2 = new JTextField();
        textField_2.setBounds(144, 164, 114, 20);
        textField_2.getDocument().addDocumentListener(new DocumentListene
r(){
               @Override
                public void insertUpdate(DocumentEvent e) {
                   cost=ComputeCost();
                }

                @Override
                public void removeUpdate(DocumentEvent e) {
                    cost=ComputeCost();
                }
                @Override
                public void changedUpdate(DocumentEvent e) {
                    cost=ComputeCost();
                }
        });
        contentPane.add(textField_2);
        textField_2.setColumns(10);

        JButton btnNewButton = new JButton("Book");
        btnNewButton.addActionListener(new ActionListener() {
```

Software Design Specification

```
            /**
             *to action listener with the buttons
             */
            @Override
            public void actionPerformed(ActionEvent e) {
                if(pgnd==null)
                    return;
                slot.setPlaygroundID(pgnd.getPlaygroundID());
                slot.setPlayerID(per.getID());
                SimpleDateFormat df=new SimpleDateFormat("dd/MM/yyyy");
                df.setLenient(false);
                try {
                slot.setSlotDate(df.parse(textField.getText()+" "+textFie
ld_1.getText()));
                }catch(Exception er)
                {
                    JOptionPane.showMessageDialog(null,er.getMessage());
                    textField.requestFocusInWindow();
                    return;
                }
                BookingSchedule bs=new BookingSchedule();
                bs.addSlot(slot);
                bs.storeSchedule();
                per.removeFromWallet(cost);
                int ownerid=pgnd.getPlaygroundOwnerID();
                Persons pers=new Persons();
                pers.findPersonByID(ownerid).addToWallet(cost);
                pers.storePersons();
                JOptionPane.showMessageDialog(null,"You've booked success
fully");
                dispose();
            }

        });
        btnNewButton.setBounds(230, 258, 89, 23);
        contentPane.add(btnNewButton);

        JButton btnNewButton_1 = new JButton("Cancel");
        btnNewButton_1.setBounds(20, 258, 89, 23);
```

```
        contentPane.add(btnNewButton_1);

        JLabel lblNewLabel_4 = new JLabel("Total Price");
        lblNewLabel_4.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblNewLabel_4.setBounds(20, 207, 78, 14);
        contentPane.add(lblNewLabel_4);

        textField_3 = new JTextField();
        textField_3.setBounds(144, 206, 114, 20);
        contentPane.add(textField_3);
        textField_3.setColumns(10);
    }
}
```

Class: BookingScheduleForm

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableColumnModel;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableColumn;

import persons.Person;
import persons.Persons;
import playgrounds.BookingSchedule;
import playgrounds.BookingSlot;
import playgrounds.Playground;
import playgrounds.Playgrounds;

import javax.swing.JLabel;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Vector;

import javax.swing.JComboBox;
```

Software Design Specification

```
import javax.swing.JTable;
import javax.swing.JButton;
/**
 * this class for Booking Schedule Form and it inherit from frame
 * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 */
@SuppressWarnings("serial")
public class BookingScheduleForm extends JFrame {
/**
 * JPanel to display all components on a frame
 */
    private JPanel contentPane;
    /**
     * details of BookingScheduleForm
     */
    String[] header= {"Date","Start time","End Time","Player Name"};
    private DefaultTableModel model=new DefaultTableModel();
    /**
     * table to show BookingScheduleForm
     */
    private JTable table=new JTable(model);

    /**
     * function to Create the frame of BookingSchedule
     * @param per is an object from class Person
     */
    public BookingScheduleForm(Person per) {
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNewLabel = new JLabel("Playgrounds");
        lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel.setBounds(10, 23, 82, 14);
```

Software Design Specification

```
        contentPane.add(lblNewLabel);

        JComboBox<String> comboBox = new JComboBox<String>();
        Vector<Integer> indeces=new Vector<Integer>();
        Playgrounds plst=new Playgrounds();
        for(int i=0;i<plst.size();i++)
        {
            Playground pgnd=plst.get(i);
            if(pgnd.getPlaygroundOwnerID()==per.getID())
            {
                comboBox.addItem(pgnd.getPlaygroundName());
                indeces.addElement(i);
            }
        }
        comboBox.setFont(new Font("Tahoma", Font.PLAIN, 13));
        comboBox.setBounds(119, 22, 146, 20);
        contentPane.add(comboBox);

        table.setBounds(29, 97, 342, 153);
        model.addColumn("Date");
        model.addColumn("Start time");
        model.addColumn("End time");
        model.addColumn("Player name");
        contentPane.add(table);

        JButton btnNewButton = new JButton("Show schedule");
        btnNewButton.setBounds(289, 21, 108, 23);
        btnNewButton.addActionListener(new ActionListener() {
        BookingSchedule sc=new BookingSchedule();
        Persons pers=new Persons();
        /**
         * to action listener with the buttons
         */
            @Override
            public void actionPerformed(ActionEvent e) {
                if(comboBox.getItemCount()==0)return;
                Playground pgnd=plst.get(indeces.elementAt(comboBox.getSe
lectedIndex()));
```

Software Design Specification

```
                for(BookingSlot slot:sc.getSlots())
                {
                    if(slot.getPlaygroundID()==pgnd.getPlaygroundID())
                    {
                        Vector<String> dat=new Vector<String>();
                        dat.add(String.valueOf(slot.getSlotDate()));
                        dat.add(String.valueOf(slot.getStartTime()));
                        dat.add(String.valueOf(slot.getEndTime()));
                        Person per=pers.findPersonByID(slot.getPlayerID()
);
                        dat.add(per.getName());
                        model.addRow(dat);
                    }
                }
            }

        });
        contentPane.add(btnNewButton);
    }
}
```

Class: CheckBalance

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import persons.Person;

import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JTextField;
/**
 * class to CheckBalance of player or playground owner
 * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
```

Software Design Specification

```
 *
 */
@SuppressWarnings("serial")
public class CheckBalance extends JFrame {
/**
 * JPanel to display all components on a frame
 */
    private JPanel contentPane;
    /**
     * JTextField to enter text
     */
    private JTextField textField;

    /**
     * function to Create the frame of CheckBalance of ewallet it appear
my balance
     * @param per is an object from class Person
     */
    public CheckBalance(Person per) {
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 369, 260);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNewLabel = new JLabel("Your balance is :");
        lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblNewLabel.setBounds(43, 99, 109, 14);
        contentPane.add(lblNewLabel);

        textField = new JTextField();
        textField.setBounds(174, 98, 86, 20);
        contentPane.add(textField);
        textField.setColumns(10);
        textField.setText(String.valueOf(per.getEwalletAmount()));
    }
}
```

Software Design Specification

Class: ConstructTeam

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import persons.Person;
import persons.Persons;
import persons.Team;
import persons.Teams;

import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.JComboBox;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.util.Vector;
import java.awt.event.ActionEvent;

@SuppressWarnings("serial")
/**
 * class to ConstructTeam
 *Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */
public class ConstructTeam extends JFrame {
/**
 * JPanel to display all components on a frame
 */
    private JPanel contentPane;
    /**
```

Software Design Specification

```
     * JTextField to enter text
     */
    private JTextField textField;
    /**
     * there is  a  relation list and listmodel ,in listmodel it show new
 added Player
     */
    DefaultListModel<String> listmodel = new DefaultListModel<String>();
    /**
     * list to show constructed team and when we click on any team it sho
w its details
     */
    private JList<String> list = new JList<String>(listmodel);
    /**
     * team is an object from class Team to construct team
     */
    private Team team=new Team();
    /**
     * function to Create the frame of construct team by adding players w
ho isn't a member in other teams
     *
     * @param per is an object from class Person
     */
    public ConstructTeam(Person per) {
        setTitle("Construct Team");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 450, 337);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNewLabel = new JLabel("Team Name");
        lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel.setBounds(26, 11, 73, 17);
        contentPane.add(lblNewLabel);

        textField = new JTextField();
        textField.setBounds(113, 11, 166, 20);
```

Software Design Specification

```
        contentPane.add(textField);
        textField.setColumns(10);

        JLabel lblNewLabel_1 = new JLabel("Player");
        lblNewLabel_1.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel_1.setBounds(20, 54, 63, 17);
        contentPane.add(lblNewLabel_1);

        Persons lst=new Persons();
        Vector<Integer> persIDs=new Vector<Integer>();
        Vector<String> persNames=new Vector<String>();
        for(int i=0;i<lst.size();i++)
        {
            Person p=lst.get(i);
            if(p.getRole()==Person.Role.PLAYER)
                {
                persNames.add(p.getName());
                persIDs.add(p.getID());
                }
        }
        JComboBox<String> comboBox = new JComboBox<String>(persNames);
        comboBox.setFont(new Font("Tahoma", Font.PLAIN, 13));
        comboBox.setBounds(82, 53, 159, 20);
        contentPane.add(comboBox);

        JLabel lblNewLabel_2 = new JLabel("Team members");
        lblNewLabel_2.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel_2.setBounds(21, 87, 100, 14);
        contentPane.add(lblNewLabel_2);

        list.setBounds(21, 112, 152, 175);
        contentPane.add(list);

        JButton btnNewButton = new JButton("Save");
        btnNewButton.setBounds(301, 264, 89, 23);
        btnNewButton.addActionListener(new ActionListener() {
/**
 * to action listener with the buttons
 */
```

Software Design Specification

```
            @Override
            public void actionPerformed(ActionEvent e) {
                Teams teams=new Teams();
                int n=teams.size();
                if(n==0)
                {
                    team.setID(1);
                }
                else
                {
                    team.setID(teams.size()+1);
                }
                team.setName(textField.getText());
                teams.add(team);
                teams.storeTeams();
                JOptionPane.showMessageDialog(null, "Team is stored succe
ssfully");
                dispose();
            }

        });
        contentPane.add(btnNewButton);
        team.addPlayer(per.getID());
        listmodel.addElement(per.getName());

        JButton btnNewButton_1 = new JButton("Add member");
        btnNewButton_1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int id=persIDs.get(comboBox.getSelectedIndex());
                if(!team.isAMember(id))
                    {
                    team.addPlayer(id);
                    listmodel.addElement((String)comboBox.getSelectedItem
());
                    }else
                    {
                        JOptionPane.showMessageDialog(null, "Player is al
ready added");
                    }
```

Software Design Specification

```
                    }
        });
        btnNewButton_1.setBounds(251, 52, 93, 23);
        contentPane.add(btnNewButton_1);

        JButton btnNewButton_2 = new JButton("Remove member");
        btnNewButton_2.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if(list.isSelectionEmpty())return;
                listmodel.remove(list.getSelectedIndex());
            }

        }
                );
        btnNewButton_2.setBounds(212, 164, 132, 23);
        contentPane.add(btnNewButton_2);
    }
}
```

Class: eWalletComponent

```
public class eWalletComponent {
}
```

Class: GofoMain

```

import java.awt.Color;

import java.awt.EventQueue;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.Window;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

Software Design Specification

```
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;

import persons.Person;
import persons.Persons;
/**
 * this is first page of program and sign in
 * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 */

@SuppressWarnings("serial")

public class GofoMain extends JFrame implements ActionListener {

    /**
     * this is a function GofoMain
     * @param args array of string arguments
     *
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        new GofoMain();
    }

/**
 * grid system of the page
 */
```

Software Design Specification

```
    JPanel pan=new JPanel(new GridBagLayout());
    /**
     * label for email and TextField to enter email
     */

    JLabel lbl1=new JLabel ("Email: ");
    JTextField tf1=new JTextField (20); //the user enter the email
    /**
     * label for Password and TextField to enter Password
     */
    JLabel lbl2=new JLabel ("Password: ");
    JPasswordField tf2=new JPasswordField (20);//the user enter the passw
ord
    /**
     * buttons for sign in and sign up and cancel
     */
    JButton btn1=new JButton("Cancel");
    JButton btn2=new JButton("Sign in");
    JButton btn3=new JButton("Sign up");
    /**
     * function to addAdmin of program
     */
    void  addAdminIfNotThere()
    {
        /**
         * take object from class Persons
         */
        Persons lst=new Persons();
        boolean found=false;
        /**
         * for loop to add person who entered his role as admin in the li
st
         */
        for(Person per:lst.getList())
        {
            if(per.getRole()==Person.Role.ADMIN)
            {
                found=true;
                break;
```

Software Design Specification

```
            }
        }
        if(found)return;
        /**
         * take per as object from class Person
         */
        Person per=new Person();
        try {
            per.setEmail("GofoProject2020@gmail.com");
            per.setPassword("admin");
            per.setRole(Person.Role.ADMIN);
            lst.add(per);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        lst.storePersons();
    }

    /**
     * for creation the page of sign  with gui
     */
    public GofoMain()
    {
        pan.setBackground(Color.lightGray);
        btn1.addActionListener(this);
        btn2.addActionListener(this);
        btn3.addActionListener(this);
        GridBagConstraints constraints = new GridBagConstraints();
        constraints.anchor = GridBagConstraints.WEST;
        constraints.insets = new Insets(10, 10, 10, 10);
        /**
         * to add the labels and PasswordField and buttons to the frame o
r page
         */
        constraints.gridx = 0;
        constraints.gridy = 0;
        pan.add(lbl1, constraints);
        constraints.gridx = 1;
```

Software Design Specification

```
        pan.add(tf1, constraints);
       constraints.gridx = 0;
        constraints.gridy = 1;
        pan.add(lbl2, constraints);
        constraints.gridx = 1;
        pan.add(tf2, constraints);
        constraints.gridx = 0;
        constraints.gridy = 2;
//        constraints.gridwidth = 2;
       constraints.anchor = GridBagConstraints.CENTER;
        pan.add(btn1, constraints);
        constraints.gridx = 1;
 //       constraints.gridwidth = 2;
        constraints.anchor = GridBagConstraints.CENTER;
        pan.add(btn2, constraints);
        constraints.gridx = 2;
 //       constraints.gridwidth = 2;
        constraints.anchor = GridBagConstraints.CENTER;
        pan.add(btn3, constraints);


        /**
         *  set border for the panel
         */
        pan.setBorder(BorderFactory.createTitledBorder(
                BorderFactory.createEtchedBorder(), "Gofo Login"));
        this.add(pan);
        pack();
        setLocationRelativeTo(null);
        /**
         * to visible the sign in page for user
         */
        this.setVisible(true);
        addAdminIfNotThere();
    }

    /**
     *  to action listener with the buttons
     */
```

Software Design Specification

```
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        if(e.getSource().equals(btn1)) //if the user press cancel close t
he sign in page
        {
             JComponent comp = (JComponent) e.getSource();
              Window win = SwingUtilities.getWindowAncestor(comp);
              win.dispose();
        }
        /**
         * if the user press sign in the user added to the list of person
s
         */

        else if(e.getSource().equals(btn2))
        {

            Persons lst=new Persons();
            Person per=lst.findPersonByEmail(tf1.getText(), String.valueO
f(tf2.getPassword()));
            lst.deletearr();
            if(per==null)
                JOptionPane.showMessageDialog(null, "Error in user name o
r password", "Error " , JOptionPane.INFORMATION_MESSAGE);

            else {
                 /**
                  * if the role is player
                  */
                    if(per.getRole()==Person.Role.PLAYER)
                    new PlayerMenu(per);
                /**
                 * if the role is playground owner
                 */
                else if(per.getRole()==Person.Role.PLAYGROUNDOWNER)
                    new PlaygroundOwnerMenu(per);
                else if(per.getRole()==Person.Role.ADMIN)
                {
```

Software Design Specification

```
                    Administrator admin=new Administrator(per);
                    admin.setVisible(true);
                }
            }
    }

        /**
         * if the user press sign up the signup is opened
         */
        else if(e.getSource().equals(btn3))
        {
            EventQueue.invokeLater(new Runnable() {
                public void run() {
                    try {
                        new SignUp();
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            });
        }
    }
}
```

Class: IeWallet

```
/**
 * this class for IeWallet of Persons
 * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 *
 */
public class IeWallet {

    /**
```

Software Design Specification

```
     * function to addToWallet
     * @param accNo is an attribute for account number of Person
     * @param amount is an attribute of amount of Person he entered
     */
    public boolean addToWallet(int accNo, double amount) {
        // TODO - implement IeWallet.addToWallet
        throw new UnsupportedOperationException();
    }

    /**
     * function to  removeFromWallet
     * @param accNo is an attribute for account number of Person
     * @param amount is an attribute of amount of Person he entered
     */
    public boolean removeFromWallet(int accNo, double amount) {
        // TODO - implement IeWallet.removeFromWallet
        throw new UnsupportedOperationException();
    }

    /**
     * function to transfer money that Person added to ewallet from his e
walllet to ewallet of playground owner
     * @param w1 is an attribute for ewallet of Person
     * @param w2  is an attribute for ewallet of Playground owner
     * @param amount is an amount of Person he added
     */
    public boolean transfere(int w1, int w2, double amount) {
        // TODO - implement IeWallet.transfere
        throw new UnsupportedOperationException();
    }

}
```

Class: InviteTeam

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
```

Software Design Specification

```
import persons.Person;
import persons.Persons;
import persons.Team;
import persons.Teams;

import javax.swing.JLabel;
import java.awt.Font;
import java.util.Vector;

import javax.swing.JComboBox;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import java.awt.event.ItemListener;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
/**
 * this is class to InviteTeam
 * * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */
@SuppressWarnings("serial")
public class InviteTeam extends JFrame {
/**
 * JPanel to display all components on a frame
 */
    private JPanel contentPane;
    /**
     * there is  a  relation list and listmodel ,in listmodel it show new
 added Player in invited team
     */
    DefaultListModel<String> listmodel=new DefaultListModel<String>();
    /**
     *
```

Software Design Specification

```
     * list to show invited team and when we click on any invited team it
show its details
     */
    JList<String> list = new JList<String>(listmodel);
    /**
     * team is an object from class Team and sets to null
     */
    Team team=null;
    /**
     * vector to store emails of invited team
     */
    Vector<String> emails=new Vector<String>();

    /**
     * function to  Create the frame of InviteTeam and i can't do it if i
don't construct team
     * @param per is an object from class Person
     */
    public InviteTeam(Person per) {
        setTitle("Invite Team");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 365, 300);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNewLabel = new JLabel("Team:");
        lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel.setBounds(10, 74, 65, 14);
        contentPane.add(lblNewLabel);

        Teams teams=new Teams();
        Vector<String> tmp = new Vector<String>();
        Vector<Team>   perteams=new Vector<Team>();
        for(int i=0;i< teams.getList().size();i++)
        {
            Team team=teams.get(i);
            if(team.getMembers().elementAt(0)==per.getID())
```

Software Design Specification

```
            {
                tmp.addElement(team.getName());
                perteams.add(team);
            }
        }
        Persons pers=new Persons();
        if(perteams.size()==1)
        {
            team=perteams.get(0);
            emails.clear();
            for(int id:team.getMembers())
            {
                Person per1=pers.findPersonByID(id);
                listmodel.addElement(per1.getName());
                emails.add(per1.getEmail());
            }
        }
        JComboBox<String> comboBox = new JComboBox<String>(tmp);
        comboBox.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                team=perteams.get(comboBox.getSelectedIndex());
                emails.clear();
                for(int id:team.getMembers())
                {
                    Person per1=pers.findPersonByID(id);
                    listmodel.addElement(per1.getName());
                    emails.add(per1.getEmail());
                }

            }
        });
        comboBox.setFont(new Font("Tahoma", Font.PLAIN, 13));
        comboBox.setBounds(56, 72, 134, 20);
        contentPane.add(comboBox);

        JLabel lblNewLabel_1 = new JLabel("Members");
        lblNewLabel_1.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel_1.setBounds(223, 23, 84, 14);
        contentPane.add(lblNewLabel_1);
```

Software Design Specification

```
        list.setBounds(223, 48, 116, 172);
        contentPane.add(list);

        JButton btnNewButton = new JButton("Invite");
        btnNewButton.setBounds(48, 197, 89, 23);
        btnNewButton.addActionListener(new ActionListener()
                {
/**
 * to action listener with the buttons
 */
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        // TODO Auto-generated method stub
                        if(team!=null)
                        {
                            String msg=per.getName() + " invites you to p
lay with him";
                            MailUtil.sendEmail(emails, "Gofo Invitation",
 msg);
                            JOptionPane.showMessageDialog(null, "An invit
ation is sent to the team");
                            dispose();
                        }

                    }

                }

                );

        contentPane.add(btnNewButton);
    }

}
```

Class: MailUtil

Software Design Specification

```
import java.util.Properties;
import java.util.Vector;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
/**
 * this class to send mails to persons with verification code
 * * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */
public class MailUtil {
        /**
         * this function  to send email with verification code and admin
send that email to Person who SignUp
         * @param to email's Person who SignUP
         * @param subject subject of email
         * @param body body of email
         */
        public static void sendEmail(String to, String subject, String bo
dy){
                String from = "GofoProject2020@gmail.com";
                final String password = "Eaalr?1999";

         Properties props = new Properties();
         props.put("mail.smtp.host", "smtp.gmail.com");
         props.put("mail.smtp.socketFactory.port", "465");
         props.put("mail.smtp.socketFactory.class",
                   "javax.net.ssl.SSLSocketFactory");
         props.put("mail.smtp.auth", "true");
         props.put("mail.smtp.port", "465");
         //get Session
         Session session = Session.getDefaultInstance(props,
```

Software Design Specification

```
            new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {

            return new PasswordAuthentication(from,password);
            }
           });
           /**
            * compose message
            */
           try {
            MimeMessage message = new MimeMessage(session);
            message.addRecipient(Message.RecipientType.TO,new InternetAddre
ss(to));
            message.setSubject(subject);
            message.setText(body);
            //send message
            Transport.send(message);
            System.out.println("message sent successfully");
           } catch (MessagingException e) {throw new RuntimeException(e);}

}
         /**
          *  function to send emails to Persons like when we invite team ,
admin send emails to invited team
          * @param to  Persons to send email
          * @param subject is the subject of email
          * @param body is the body of email
          */
         public static void sendEmail(Vector<String> to, String subject, S
tring body){
                  String from = "GofoProject2020@gmail.com";
                  final String password = "Eaalr?1999";

         Properties props = new Properties();
         props.put("mail.smtp.host", "smtp.gmail.com");
         props.put("mail.smtp.socketFactory.port", "465");
         props.put("mail.smtp.socketFactory.class",
                   "javax.net.ssl.SSLSocketFactory");
         props.put("mail.smtp.auth", "true");
```

Software Design Specification

```
        props.put("mail.smtp.port", "465");
        /**
         * get Session
         */
        Session session = Session.getDefaultInstance(props,
         new javax.mail.Authenticator() {
         protected PasswordAuthentication getPasswordAuthentication() {

         return new PasswordAuthentication(from,password);
         }
        });
        /**
         * compose message
         */
        try {
         MimeMessage message = new MimeMessage(session);
         for(String tto:to) message.addRecipient(Message.RecipientType.TO
,new InternetAddress(tto));
         message.setSubject(subject);
         message.setText(body);
         /**
          * send message
          */
         Transport.send(message);
         System.out.println("message sent successfully");
        } catch (MessagingException e) {throw new RuntimeException(e);}

}
}
```

Class: PlayerMenu

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

Software Design Specification

```
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

import persons.Person;
/**
 * class to show PlayerMenu and he can do what he want
 * * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */
public class PlayerMenu implements ActionListener{
/**
 * label for say Hello Player and his name
 */
public JLabel label;
/**
 * object from class Person
 */
private Person per;
private JButton [][] buttons=new JButton[6][1];
/**
 * function to show PlayerMenu and he can do what he want
 * @param per1 is an object from class Person and his role is Player
 */
    public PlayerMenu(Person per1){
        per=per1;
        JFrame frame=new JFrame();
        JLabel label=new JLabel("Hello Player: " + per.getName());
        JPanel panel=new JPanel();
        JPanel panel2=new JPanel();
        JPanel panel3=new JPanel();
        panel.add(label);
```

Software Design Specification

```
        String  [][]player_function= {{"Update_Profile"},{"Show_playGroun
ds"},{"Book_playground"},{"Check_eWallet"},{"Invite_Team"},{"Create_Team"
}};
        String[]BE= {"Back","Exit"};
        JButton[]buttons2=new JButton[2];
        for(int i=0 ;i<6;i++)
        {
            for(int j=0 ;j<1;j++)
            {
                buttons[i][j]=new JButton(player_function[i][j]);
                buttons[i][j].setBackground(Color.LIGHT_GRAY);
                buttons[i][j].setForeground(Color.GRAY);
                buttons[i][j].addActionListener(this);
                panel2.add(buttons[i][j]);
            }
        }
        for(int i=0 ;i<2;i++)
        {
            buttons2[i]=new JButton(BE[i]);

            buttons2[i].setBackground(Color.DARK_GRAY);
            buttons2[i].setForeground(Color.BLUE);
            panel3.add(buttons2[i]);
        }

        panel.setBorder(BorderFactory.createEmptyBorder(20,20,10,20));
        panel.setLayout(new GridLayout(1,1));

        panel2.setBorder(BorderFactory.createEmptyBorder(40,20,20,20));
        panel2.setLayout(new GridLayout(4,2));


        frame.add(panel,BorderLayout.CENTER);
        frame.add(panel2,BorderLayout.AFTER_LAST_LINE);
        frame.add(panel3,BorderLayout.AFTER_LINE_ENDS);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("GOFO_PLAYER");
        frame.pack();
        frame.setVisible(true);
```

Software Design Specification

```
    }
    /**
     * to action listener with the buttons
     */
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        JButton btn=(JButton)e.getSource();
        if(btn.equals(buttons[0][0]))
        {
            new SignUp(per);
        }else if(btn.equals(buttons[1][0]))
        {
            new ShowPlaygrounds().setVisible(true);
        }else if(btn.equals(buttons[2][0]))
        {
            new BookingPlayground(per).setVisible(true);
        }
        else if(btn.equals(buttons[3][0]))
        {
        new CheckBalance(per).setVisible(true);
        }
        else if(btn.equals(buttons[4][0]))
        {
            new InviteTeam(per).setVisible(true);
        }else if(btn.equals(buttons[5][0]))
        {
            new ConstructTeam(per).setVisible(true);
        }


    }


    }
```

Software Design Specification

Class: PlaygroundForm

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import persons.Person;
import playgrounds.Playground;
import playgrounds.Playgrounds;
import playgrounds.AvailabilityStatus;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.SwingConstants;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JRadioButton;
/**
 * class for PlaygroundForm
 ** Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */
@SuppressWarnings("serial")
public class PlaygroundForm extends JFrame implements ActionListener{
/**
 * JPanel to display all components on a frame
 */
    private JPanel contentPane;
    /**
     * JTextField to enter text
     */
    private JTextField textField;
    private JTextField textField_1;
```

Software Design Specification

```
    private JTextField textField_2;
    private JTextField textField_3;
    private JTextField textField_4;
    /**
     * JRadioButton to show the availability status of playground and you
 select from it
     */
    private JRadioButton rdbtnNewRadioButton;
    private JRadioButton rdbtnNewRadioButton_1;
    private JRadioButton rdbtnNewRadioButton_2;
    private JTextField textField_6;
    private Playground pgnd=new Playground();
    private JButton btnNewButton;
    private JButton btnNewButton_1;
    private Person per;

    /**
     * function to Create the frame of PlaygroundForm and entered playgro
und details
     * @param per is an object from class Person
     */
public PlaygroundForm(Person per) {
        this.per=per;
        setTitle("Playground");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 467, 383);
        contentPane = new JPanel();
        contentPane.setToolTipText("");
        contentPane.setBorder(new EmptyBorder(18, 10, 10, 10));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNewLabel = new JLabel("Playground Name :");
        lblNewLabel.setBounds(10, 24, 108, 17);
        lblNewLabel.setHorizontalAlignment(SwingConstants.LEFT);
        lblNewLabel.setFont(new Font("Times New Roman", Font.PLAIN, 14));
        contentPane.add(lblNewLabel);

        textField = new JTextField();
```

Software Design Specification

```
        textField.setBounds(148, 23, 204, 20);
        contentPane.add(textField);
        textField.setColumns(10);

        JLabel lblNewLabel_1 = new JLabel("Playground ID :");
        lblNewLabel_1.setFont(new Font("Times New Roman", Font.PLAIN, 14)
);
        lblNewLabel_1.setBounds(10, 52, 108, 14);
        contentPane.add(lblNewLabel_1);

        textField_1 = new JTextField();
        textField_1.setBounds(148, 54, 204, 20);
        contentPane.add(textField_1);
        textField_1.setColumns(10);

        JLabel lblNewLabel_2 = new JLabel("Playground Location :");
        lblNewLabel_2.setFont(new Font("Times New Roman", Font.PLAIN, 14)
);
        lblNewLabel_2.setBounds(10, 83, 124, 14);
        contentPane.add(lblNewLabel_2);

        textField_2 = new JTextField();
        textField_2.setBounds(148, 81, 204, 20);
        contentPane.add(textField_2);
        textField_2.setColumns(10);

        JLabel lblNewLabel_3 = new JLabel("Playground Size :");
        lblNewLabel_3.setFont(new Font("Times New Roman", Font.PLAIN, 14)
);
        lblNewLabel_3.setBounds(10, 108, 108, 14);
        contentPane.add(lblNewLabel_3);

        textField_3 = new JTextField();
        textField_3.setBounds(148, 106, 204, 20);
        contentPane.add(textField_3);
        textField_3.setColumns(10);

        JLabel lblNewLabel_4 = new JLabel("Playground Hours :");
```

Software Design Specification

```
        lblNewLabel_4.setFont(new Font("Times New Roman", Font.PLAIN, 14)
);
        lblNewLabel_4.setBounds(10, 133, 124, 14);
        contentPane.add(lblNewLabel_4);

        textField_4 = new JTextField();
        textField_4.setBounds(148, 131, 204, 20);
        contentPane.add(textField_4);
        textField_4.setColumns(10);

        JLabel lblNewLabel_5 = new JLabel("Playground Availability :");
        lblNewLabel_5.setFont(new Font("Times New Roman", Font.PLAIN, 14)
);
        lblNewLabel_5.setBounds(10, 158, 140, 14);
        contentPane.add(lblNewLabel_5);

        JLabel lblNewLabel_6 = new JLabel("Playground Price / hour:");
        lblNewLabel_6.setFont(new Font("Times New Roman", Font.PLAIN, 14)
);
        lblNewLabel_6.setBounds(10, 185, 140, 14);
        contentPane.add(lblNewLabel_6);

        textField_6 = new JTextField();
        textField_6.setBounds(148, 183, 204, 20);
        contentPane.add(textField_6);
        textField_6.setColumns(10);

        btnNewButton = new JButton("Save");
        btnNewButton.setBounds(333, 310, 89, 23);
        btnNewButton.addActionListener(this);
        contentPane.add(btnNewButton);

        btnNewButton_1 = new JButton("Cancel");
        btnNewButton_1.setBounds(29, 310, 89, 23);
        btnNewButton_1.addActionListener(this);
        contentPane.add(btnNewButton_1);

        rdbtnNewRadioButton = new JRadioButton("Suspended");
        rdbtnNewRadioButton.setBounds(148, 155, 81, 23);
```

Software Design Specification

```
        contentPane.add(rdbtnNewRadioButton);

        rdbtnNewRadioButton_1 = new JRadioButton("Active");
        rdbtnNewRadioButton_1.setBounds(231, 155, 55, 23);
        contentPane.add(rdbtnNewRadioButton_1);

        rdbtnNewRadioButton_2 = new JRadioButton("Deleted");
        rdbtnNewRadioButton_2.setBounds(284, 155, 109, 23);
        contentPane.add(rdbtnNewRadioButton_2);
    }
/**
 * function to select the availability status of playground
 */
private void mapFromPlaygraound()
{
    textField.setText(pgnd.getPlaygroundName());
    textField_1.setText(String.valueOf(pgnd.getPlaygroundID()));
    textField_2.setText(pgnd.getPlaygroundLocation());
    textField_3.setText(String.valueOf(pgnd.getPlaygroundSize()));
    textField_4.setText(String.valueOf(pgnd.getPlaygroundHours()));
    switch(pgnd.getPlaygroundAvailbility())
    {
    case ACTIVE:
        rdbtnNewRadioButton.setSelected(true);
        rdbtnNewRadioButton_1.setSelected(false);
        rdbtnNewRadioButton_2.setSelected(false);
        break;
    case SUSPENDED:
        rdbtnNewRadioButton.setSelected(false);
        rdbtnNewRadioButton_1.setSelected(true);
        rdbtnNewRadioButton_2.setSelected(false);
        break;
    case DELETED:
        rdbtnNewRadioButton.setSelected(false);
        rdbtnNewRadioButton_1.setSelected(false);
        rdbtnNewRadioButton_2.setSelected(true);
        break;
    }
    textField_6.setText(String.valueOf(pgnd.getPlaygroundPrice()));
```

Software Design Specification

```
}
/**
 * function to show what i entered to add playground
 */
private void mapToPlayground()
{
    pgnd.setPlaygroundName(textField.getText());
    pgnd.setPlaygroundID(Integer.parseInt(textField_1.getText()));
    pgnd.setPlaygroundLocation(textField_2.getText());
    pgnd.setPlaygroundSize(Double.parseDouble(textField_3.getText()));
    pgnd.setPlaygroundHours(Integer.parseInt(textField_4.getText()));
    rdbtnNewRadioButton.setSelected(false);rdbtnNewRadioButton.setEnabled
(false);
    rdbtnNewRadioButton_1.setSelected(true);rdbtnNewRadioButton_1.setEnab
led(false);
    rdbtnNewRadioButton_2.setSelected(false);rdbtnNewRadioButton_2.setEna
bled(false);

    if(rdbtnNewRadioButton.isSelected())
        pgnd.setPlaygroundAvailbility(AvailabilityStatus.ACTIVE);
    else if(rdbtnNewRadioButton_1.isSelected())
        pgnd.setPlaygroundAvailbility(AvailabilityStatus.SUSPENDED);
    else
        pgnd.setPlaygroundAvailbility(AvailabilityStatus.DELETED);
    pgnd.setPlaygroundPrice(Double.parseDouble(textField_6.getText()));
    pgnd.setPlaygroundOwnerID(per.getID());
}
/**
 * to action listener with the buttons
 */
@Override
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    JButton btn=(JButton)e.getSource();
    if(btnNewButton.equals(btn))
    {
        mapToPlayground();
        Playgrounds lst=new Playgrounds();
        if(lst.size()==0)
```

Software Design Specification

```
            lst.add(pgnd);
        else {
            int i=lst.findPlaygroundByID(pgnd.getPlaygroundID());
            if(i==lst.size())lst.add(pgnd);else lst.set(i, pgnd);
        }
        lst.storePlaygrounds();
        JOptionPane.showMessageDialog(null, "You've successfully added a
Playground");
        this.dispose();
    }else if(btnNewButton_1.equals(btn))
    {
        this.dispose();
    }
}
}
```

Class: PlaygroundOwnerMenu

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

import persons.Person;
/**
 * class of PlaygroundOwnerMenu
 *  Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
```

Software Design Specification

```
 *
 */
public class PlaygroundOwnerMenu implements ActionListener{
private JButton [][] buttons=new JButton[4][1];
public JLabel label;
Person per;
/**
 * function to show PlaygroundOwnerMenu and you can select what you want
to do
 * @param per1 is an object from class Person
 */
    public PlaygroundOwnerMenu(Person per1){
        per=per1;
        JFrame frame=new JFrame();
        JLabel label=new JLabel("Hello Playground owner: "+per.getName())
;
        JPanel panel=new JPanel();
        JPanel panel2=new JPanel();
        JPanel panel3=new JPanel();
        panel.add(label);
        String  [][]player_function= {{"Update_Profile"},{"Add_Playground
"},{"Check_ewallet"},{"Booking Schedule"}};
        String[]BE= {"Back","Exit"};
        JButton[]buttons2=new JButton[2];
        for(int i=0 ;i<4;i++)
        {
            for(int j=0 ;j<1;j++)
            {
                buttons[i][j]=new JButton(player_function[i][j]);
                buttons[i][j].addActionListener(this);
                buttons[i][j].setBackground(Color.LIGHT_GRAY);
                buttons[i][j].setForeground(Color.GRAY);
                panel2.add(buttons[i][j]);
            }
        }
        for(int i=0 ;i<2;i++)
        {
            buttons2[i]=new JButton(BE[i]);
```

Software Design Specification

```
            buttons2[i].setBackground(Color.DARK_GRAY);
            buttons2[i].setForeground(Color.BLUE);
            panel3.add(buttons2[i]);
        }

        panel.setBorder(BorderFactory.createEmptyBorder(20,20,10,20));
        panel.setLayout(new GridLayout(1,1));

        panel2.setBorder(BorderFactory.createEmptyBorder(40,20,20,20));
        panel2.setLayout(new GridLayout(4,2));


        frame.add(panel,BorderLayout.CENTER);
        frame.add(panel2,BorderLayout.AFTER_LAST_LINE);
        frame.add(panel3,BorderLayout.AFTER_LINE_ENDS);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("GOFO_PLAYER");
        frame.pack();
        frame.setVisible(true);

    }
/**
 * to action listener with the buttons
 */
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        JButton btn=(JButton)e.getSource();
        if(btn.equals(buttons[0][0]))
        {
            new SignUp(per);
        }else if(btn.equals(buttons[1][0]))
        {
            new PlaygroundForm(per).setVisible(true);
        }else if(btn.equals(buttons[2][0]))
        {
            new CheckBalance(per).setVisible(true);
        }else if(btn.equals(buttons[3][0]))
        {
```

Software Design Specification

```
            new BookingScheduleForm(per).setVisible(true);
        }
    }

}
```

Class: ShowAdminApprove

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Vector;

import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

import playgrounds.AvailabilityStatus;
import playgrounds.Playground;
import playgrounds.Playgrounds;
/**
 * this class for approve Playground
 * * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */
public class ShowAdminApprove{

    private JFrame frame=new JFrame();
    /**
     * combo box to select from it
     */
    private JComboBox<String> combo1 ;
    /**
     * button named APProve
```

Software Design Specification

```
     */
    private JButton btn=new JButton("APProve");
    /**
     * function to approve playground that playground owner added and  th
is approve made by admin
     */

    public ShowAdminApprove()
    {
        combo1=new JComboBox<String>();
        Vector<Integer> indeces=new Vector<Integer>();
        Playgrounds play=new Playgrounds();
        for(int i=0;i<play.size();i++)
        {
            Playground playi=play.get(i);
            if(playi.getPlaygroundAvailbility().equals((AvailabilityStatu
s.SUSPENDED)))
            {
                combo1.addItem(playi.getPlaygroundName());
                indeces.add(i);
            }
        }
        frame.add(combo1);
        combo1.setBounds(100, 10, 150, 20);
        frame.add(btn);
        btn.setBounds(100, 40, 150, 20);

        btn.addActionListener(new ActionListener() {
            /**
             * to action listener with the buttons
             */
            @Override
            public void actionPerformed(ActionEvent e) {
                if(combo1.getItemCount()==0)return;
                int index=indeces.get(combo1.getSelectedIndex());
                play.get(index).setPlaygroundAvailbility(AvailabilityStat
us.ACTIVE);
                play.storePlaygrounds();
                indeces.removeElementAt(combo1.getSelectedIndex());
```

Software Design Specification

```
                combo1.removeItemAt(combo1.getSelectedIndex());
                JOptionPane.showMessageDialog(null,AvailabilityStatus.ACT
IVE ,"APProve_PLayground", JOptionPane.INFORMATION_MESSAGE);
            }
        });
        frame.setTitle("APProve_PLayground");
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setSize(400,200);
        frame.setResizable(false);
        frame.setLayout(null);
        frame.setVisible(true);
    }

}
```

Class: ShowAdminDelete

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Vector;

import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import playgrounds.AvailabilityStatus;
import playgrounds.Playground;
import playgrounds.Playgrounds;
/**
 * this class to delete playground that admin don't approve it
 * * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */
public class ShowAdminDelete {
```

Software Design Specification

```
    private JFrame frame=new JFrame();
    /**
     * combo box to select from it
     */
    private JComboBox<String> combo1 = new JComboBox<String>();
    /**
     * button named Delete
     */
    private JButton btn=new JButton("Delete");
    /**
     * function to delete not approved playgrounds
     */
    public ShowAdminDelete()
    {
        Vector<Integer> indeces=new Vector<Integer>();
        Playgrounds play=new Playgrounds();
        for(int i=0;i<play.size();i++)
        {
            Playground playi=play.get(i);
            if(playi.getPlaygroundAvailbility().equals((AvailabilityStatu
s.ACTIVE))|| playi.getPlaygroundAvailbility().equals((AvailabilityStatus.
SUSPENDED)))
            {
                combo1.addItem(playi.getPlaygroundName());
                indeces.add(i);
            }
        }
        frame.add(combo1);
        combo1.setBounds(100, 10, 150, 20);
        frame.add(btn);
        btn.setBounds(100, 40, 150, 20);
        btn.addActionListener(new ActionListener() {
            /**
             * to action listener with the buttons
             */
            @Override
            public void actionPerformed(ActionEvent e) {
                if(combo1.getItemCount()==0)return;
                int index=indeces.get(combo1.getSelectedIndex());
```

Software Design Specification

```
                play.get(index).setPlaygroundAvailbility(AvailabilityStat
us.DELETED);
                play.storePlaygrounds();
                indeces.removeElementAt(combo1.getSelectedIndex());
                combo1.removeItemAt(combo1.getSelectedIndex());
                JOptionPane.showMessageDialog(null,AvailabilityStatus.DEL
ETED ,"", JOptionPane.INFORMATION_MESSAGE);
            }
        });
        frame.setTitle("Delete_PLayground");
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setSize(400,200);
        frame.setResizable(false);
        frame.setLayout(null);
        frame.setVisible(true);
    }
}
```

Class: ShowAdminSuspend

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Vector;

import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import playgrounds.AvailabilityStatus;
import playgrounds.Playground;
import playgrounds.Playgrounds;
/**
 * this class for show suspend playgrounds
 *  Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
```

Software Design Specification

```
 *
 */
public class ShowAdminSuspend  {
    private JFrame frame=new JFrame();
    private JComboBox<String> combo1 = new JComboBox<String>();
/**
 * button named SUSPEND
 */
    private JButton btn=new JButton("SUSPEND");
    /**
     * function that show suspended playgrounds and player can't booked i
t until it available
     */
    public ShowAdminSuspend()
    {
        Vector<Integer> indeces=new Vector<Integer>();
        Playgrounds play=new Playgrounds();
        for(int i=0;i<play.size();i++)
        {
            Playground playi=play.get(i);
            if(playi.getPlaygroundAvailbility().equals((AvailabilityStatu
s.ACTIVE)))
            {
                combo1.addItem(playi.getPlaygroundName());
                indeces.add(i);
            }
        }
        frame.add(combo1);
        combo1.setBounds(100, 10, 150, 20);
        frame.add(btn);
        btn.setBounds(100, 40, 150, 20);
        btn.addActionListener(new ActionListener() {
            /**
             * to action listener with the buttons
             */
            @Override
            public void actionPerformed(ActionEvent e) {
                if(combo1.getItemCount()==0)return;
                int index=indeces.get(combo1.getSelectedIndex());
```

Software Design Specification

```
                play.get(index).setPlaygroundAvailbility(AvailabilityStat
us.SUSPENDED);
                play.storePlaygrounds();
                indeces.removeElementAt(combo1.getSelectedIndex());
                combo1.removeItemAt(combo1.getSelectedIndex());
                JOptionPane.showMessageDialog(null,AvailabilityStatus.SUS
PENDED ,"SUSPEND_PLayground", JOptionPane.INFORMATION_MESSAGE);
            }
        });
        frame.setTitle("SUSPENDED_PLayground");
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setSize(400,200);
        frame.setResizable(false);
        frame.setLayout(null);
        frame.setVisible(true);
    }
}
```

Class: ShowPlaygrounds

```
import javax.swing.DefaultListModel;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import playgrounds.AvailabilityStatus;
import playgrounds.Playground;
import playgrounds.Playgrounds;

import javax.swing.JList;
import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.ListSelectionModel;
import javax.swing.event.ListSelectionListener;

import persons.Person;
```

Software Design Specification

```
import persons.Persons;

import javax.swing.event.ListSelectionEvent;
/**
 * this class for ShowPlaygrounds informations and player can select from
 it
 * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */
@SuppressWarnings("serial")
public class ShowPlaygrounds extends JFrame {
/**
 * JPanel to display all components on a frame
 */
    private JPanel contentPane;
    /**
     * JTextField to enter text
     */
    private JTextField textField;
    private JTextField textField_1;
    private JTextField textField_2;
    private JTextField textField_3;
    /**
     * there is  a  relation list and listmodel ,in listmodel it show new
 added playground
     */
    DefaultListModel<String> listmodel=new DefaultListModel<String>();
    /**
     * list to show playgrounds and when we click on any playground it sh
ow its details
     */
    private JList<String> list = new JList<String>(listmodel);
    private JTextField textField_4;

    /**
     * function to Create the frame of ShowPlaygrounds details that playg
round owner added and player can book one from it
```

Software Design Specification

```
     */
    public ShowPlaygrounds() {
        setTitle("Show Playgrounds");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 450, 326);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        Playgrounds lst=new Playgrounds();
        for(int i=0;i<lst.size();i++)
            if(lst.get(i).getPlaygroundAvailbility()==AvailabilityStatus.
ACTIVE)
                        listmodel.addElement(lst.get(i).getPlaygroundName
());
        list.addListSelectionListener(new ListSelectionListener() {
            public void valueChanged(ListSelectionEvent e) {
                int index=list.getSelectedIndex();
                Playground pgnd=lst.get(index);
                textField.setText(String.valueOf(pgnd.getPlaygroundID()))
;
                textField_1.setText(pgnd.getPlaygroundLocation());
                textField_2.setText(String.valueOf(pgnd.getPlaygroundHour
s()));
                textField_3.setText(String.valueOf(pgnd.getPlaygroundPric
e()));
                Person per=new Persons().findPersonByID(pgnd.getPlaygroun
dOwnerID());
                textField_4.setText(per.getName());
            }
        });
        list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

        list.setBounds(10, 35, 131, 226);
        contentPane.add(list);

        JLabel lblNewLabel = new JLabel("Playgrounds");
        lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel.setBounds(26, 11, 84, 17);
```

Software Design Specification

```
        contentPane.add(lblNewLabel);

        JLabel lblNewLabel_1 = new JLabel("Playground ID");
        lblNewLabel_1.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel_1.setBounds(161, 48, 88, 17);
        contentPane.add(lblNewLabel_1);

        textField = new JTextField();
        textField.setBounds(302, 48, 110, 20);
        contentPane.add(textField);
        textField.setColumns(10);

        JLabel lblNewLabel_2 = new JLabel("Playground Location");
        lblNewLabel_2.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel_2.setBounds(161, 96, 131, 17);
        contentPane.add(lblNewLabel_2);

        textField_1 = new JTextField();
        textField_1.setBounds(302, 96, 110, 20);
        contentPane.add(textField_1);
        textField_1.setColumns(10);

        JLabel lblNewLabel_3 = new JLabel("Available Hours");
        lblNewLabel_3.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel_3.setBounds(161, 140, 131, 14);
        contentPane.add(lblNewLabel_3);

        textField_2 = new JTextField();
        textField_2.setBounds(302, 139, 110, 20);
        contentPane.add(textField_2);
        textField_2.setColumns(10);

        JLabel lblNewLabel_4 = new JLabel("Price");
        lblNewLabel_4.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel_4.setBounds(161, 186, 46, 14);
        contentPane.add(lblNewLabel_4);

        textField_3 = new JTextField();
        textField_3.setBounds(302, 185, 110, 20);
```

```
        contentPane.add(textField_3);
        textField_3.setColumns(10);

        JLabel lblNewLabel_5 = new JLabel("Playground Owner");
        lblNewLabel_5.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel_5.setBounds(161, 228, 131, 17);
        contentPane.add(lblNewLabel_5);

        textField_4 = new JTextField();
        textField_4.setBounds(301, 228, 111, 20);
        contentPane.add(textField_4);
        textField_4.setColumns(10);
    }
}
```

Class: SignUp

```
import javax.swing.JFrame;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

import java.awt.Insets;
import java.awt.Window;
import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;

import persons.Person;
import persons.Persons;

import javax.swing.JRadioButton;
import javax.swing.JButton;
import javax.swing.JComponent;

import java.awt.event.ActionListener;
```

Software Design Specification

```
import java.awt.event.ActionEvent;
/**
 * this class to SignUp
 *  Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */
public class SignUp {
    /**
     * JFrame for GofoLogin
     */
    public JFrame frmGofoLogin;
    /**
     * JTextField to enter text
     */
    private JTextField textField;
    private JTextField textField_1;
    private JLabel lblNewLabel_2;
    private JLabel lblNewLabel_3;
    private JTextField textField_2;
    private JTextField textField_3;
    private JLabel lblNewLabel_4;
    private JTextField textField_4;
    private JLabel lblNewLabel_5;
    private JTextField textField_5;
    private JLabel lblNewLabel_6;
    private JTextField textField_7;
    /**
     * JRadioButton to select role
     */
    private JRadioButton rdbtnNewRadioButton;
    private JRadioButton rdbtnNewRadioButton_1;
    private JLabel lblNewLabel_7;
    private JTextField textField_6;
    private JButton btnNewButton;
    private JButton btnNewButton_1;
    private JLabel lblNewLabel_8;
    private int mode=1;
```

Software Design Specification

```
    Person per;
    /**
     * function to Create sign up.
     */
    public SignUp() {
        mode=1;
        per=new Person();
        initialize();
    }
    /**
     * function to update entered Person's information
     * @param per is an object from class Person
     */
    public SignUp(Person per)
    {
        mode=2;
        this.per=per;
        initialize();
        mapFromPerson(per);
    }

    /**
     * function to Initialize the contents of the frame and details you w
ill entered to sign up.
     */
    private void initialize() {
        frmGofoLogin = new JFrame();
        frmGofoLogin.getContentPane().setFont(new Font("Adobe Arabic", Fo
nt.PLAIN, 15));
        frmGofoLogin.setTitle(mode==1?"Gofo SignUp":"Update Profile");
        frmGofoLogin.setBounds(100, 100, 372, 300);
        frmGofoLogin.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        GridBagLayout gridBagLayout = new GridBagLayout();
        gridBagLayout.columnWidths = new int[]{0, 0, 0, 0, 0, 0, 0};
        gridBagLayout.rowHeights = new int[]{0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0};
        gridBagLayout.columnWeights = new double[]{0.0, 1.0, 1.0, 1.0, 0.
0, 1.0, Double.MIN_VALUE};
```

Software Design Specification

```
        gridBagLayout.rowWeights = new double[]{0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, Double.MIN_VALUE};
        frmGofoLogin.getContentPane().setLayout(gridBagLayout);

        JLabel lblNewLabel = new JLabel("First Name");
        lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
        GridBagConstraints gbc_lblNewLabel = new GridBagConstraints();
        gbc_lblNewLabel.insets = new Insets(0, 0, 5, 5);
        gbc_lblNewLabel.gridx = 1;
        gbc_lblNewLabel.gridy = 1;
        frmGofoLogin.getContentPane().add(lblNewLabel, gbc_lblNewLabel);

        textField = new JTextField();
        GridBagConstraints gbc_textField = new GridBagConstraints();
        gbc_textField.fill = GridBagConstraints.HORIZONTAL;
        gbc_textField.gridwidth = 2;
        gbc_textField.insets = new Insets(0, 0, 5, 5);
        gbc_textField.gridx = 2;
        gbc_textField.gridy = 1;
        frmGofoLogin.getContentPane().add(textField, gbc_textField);
        textField.setColumns(10);

        JLabel lblNewLabel_1 = new JLabel("Last Name");
        lblNewLabel_1.setFont(new Font("Tahoma", Font.PLAIN, 14));
        GridBagConstraints gbc_lblNewLabel_1 = new GridBagConstraints();
        gbc_lblNewLabel_1.insets = new Insets(0, 0, 5, 5);
        gbc_lblNewLabel_1.gridx = 1;
        gbc_lblNewLabel_1.gridy = 2;
        frmGofoLogin.getContentPane().add(lblNewLabel_1, gbc_lblNewLabel_
1);

        textField_1 = new JTextField();
        GridBagConstraints gbc_textField_1 = new GridBagConstraints();
        gbc_textField_1.gridwidth = 2;
        gbc_textField_1.insets = new Insets(0, 0, 5, 5);
        gbc_textField_1.fill = GridBagConstraints.HORIZONTAL;
        gbc_textField_1.gridx = 2;
        gbc_textField_1.gridy = 2;
        frmGofoLogin.getContentPane().add(textField_1, gbc_textField_1);
```

Software Design Specification

```
        textField_1.setColumns(10);

        lblNewLabel_2 = new JLabel("eMail");
        lblNewLabel_2.setFont(new Font("Tahoma", Font.PLAIN, 14));
        GridBagConstraints gbc_lblNewLabel_2 = new GridBagConstraints();
        gbc_lblNewLabel_2.insets = new Insets(0, 0, 5, 5);
        gbc_lblNewLabel_2.gridx = 1;
        gbc_lblNewLabel_2.gridy = 3;
        frmGofoLogin.getContentPane().add(lblNewLabel_2, gbc_lblNewLabel_
2);

        textField_2 = new JTextField();
        GridBagConstraints gbc_textField_2 = new GridBagConstraints();
        gbc_textField_2.gridwidth = 2;
        gbc_textField_2.insets = new Insets(0, 0, 5, 5);
        gbc_textField_2.fill = GridBagConstraints.HORIZONTAL;
        gbc_textField_2.gridx = 2;
        gbc_textField_2.gridy = 3;
        frmGofoLogin.getContentPane().add(textField_2, gbc_textField_2);
        textField_2.setColumns(10);
        if(mode==2)textField_2.setEnabled(false);

        lblNewLabel_3 = new JLabel("Password\r\n");
        lblNewLabel_3.setFont(new Font("Tahoma", Font.PLAIN, 14));
        GridBagConstraints gbc_lblNewLabel_3 = new GridBagConstraints();
        gbc_lblNewLabel_3.insets = new Insets(0, 0, 5, 5);
        gbc_lblNewLabel_3.gridx = 1;
        gbc_lblNewLabel_3.gridy = 4;
        frmGofoLogin.getContentPane().add(lblNewLabel_3, gbc_lblNewLabel_
3);

        textField_3 = new JTextField();
        GridBagConstraints gbc_textField_3 = new GridBagConstraints();
        gbc_textField_3.gridwidth = 2;
        gbc_textField_3.insets = new Insets(0, 0, 5, 5);
        gbc_textField_3.fill = GridBagConstraints.HORIZONTAL;
        gbc_textField_3.gridx = 2;
        gbc_textField_3.gridy = 4;
        frmGofoLogin.getContentPane().add(textField_3, gbc_textField_3);
```

Software Design Specification

```
        textField_3.setColumns(10);

        lblNewLabel_4 = new JLabel("Phone Number");
        lblNewLabel_4.setFont(new Font("Tahoma", Font.PLAIN, 14));
        GridBagConstraints gbc_lblNewLabel_4 = new GridBagConstraints();
        gbc_lblNewLabel_4.insets = new Insets(0, 0, 5, 5);
        gbc_lblNewLabel_4.gridx = 1;
        gbc_lblNewLabel_4.gridy = 5;
        frmGofoLogin.getContentPane().add(lblNewLabel_4, gbc_lblNewLabel_
4);

        textField_4 = new JTextField();
        GridBagConstraints gbc_textField_4 = new GridBagConstraints();
        gbc_textField_4.gridwidth = 2;
        gbc_textField_4.insets = new Insets(0, 0, 5, 5);
        gbc_textField_4.fill = GridBagConstraints.HORIZONTAL;
        gbc_textField_4.gridx = 2;
        gbc_textField_4.gridy = 5;
        frmGofoLogin.getContentPane().add(textField_4, gbc_textField_4);
        textField_4.setColumns(10);

        lblNewLabel_5 = new JLabel("Location");
        lblNewLabel_5.setFont(new Font("Tahoma", Font.PLAIN, 14));
        GridBagConstraints gbc_lblNewLabel_5 = new GridBagConstraints();
        gbc_lblNewLabel_5.insets = new Insets(0, 0, 5, 5);
        gbc_lblNewLabel_5.gridx = 1;
        gbc_lblNewLabel_5.gridy = 6;
        frmGofoLogin.getContentPane().add(lblNewLabel_5, gbc_lblNewLabel_
5);

        textField_5 = new JTextField();
        GridBagConstraints gbc_textField_5 = new GridBagConstraints();
        gbc_textField_5.gridwidth = 2;
        gbc_textField_5.insets = new Insets(0, 0, 5, 5);
        gbc_textField_5.fill = GridBagConstraints.HORIZONTAL;
        gbc_textField_5.gridx = 2;
        gbc_textField_5.gridy = 6;
        frmGofoLogin.getContentPane().add(textField_5, gbc_textField_5);
        textField_5.setColumns(10);
```

Software Design Specification

```
        lblNewLabel_6 = new JLabel("Role");
        lblNewLabel_6.setFont(new Font("Tahoma", Font.PLAIN, 14));
        GridBagConstraints gbc_lblNewLabel_6 = new GridBagConstraints();
        gbc_lblNewLabel_6.insets = new Insets(0, 0, 5, 5);
        gbc_lblNewLabel_6.gridx = 1;
        gbc_lblNewLabel_6.gridy = 7;
        frmGofoLogin.getContentPane().add(lblNewLabel_6, gbc_lblNewLabel_
6);

        rdbtnNewRadioButton = new JRadioButton("Playground Owner");
        GridBagConstraints gbc_rdbtnNewRadioButton = new GridBagConstrain
ts();
        gbc_rdbtnNewRadioButton.insets = new Insets(0, 0, 5, 5);
        gbc_rdbtnNewRadioButton.gridx = 2;
        gbc_rdbtnNewRadioButton.gridy = 7;
        frmGofoLogin.getContentPane().add(rdbtnNewRadioButton, gbc_rdbtnN
ewRadioButton);

        rdbtnNewRadioButton_1 = new JRadioButton("Player");
        GridBagConstraints gbc_rdbtnNewRadioButton_1 = new GridBagConstra
ints();
        gbc_rdbtnNewRadioButton_1.insets = new Insets(0, 0, 5, 5);
        gbc_rdbtnNewRadioButton_1.gridx = 3;
        gbc_rdbtnNewRadioButton_1.gridy = 7;
        frmGofoLogin.getContentPane().add(rdbtnNewRadioButton_1, gbc_rdbt
nNewRadioButton_1);
        if(mode!=1)
        {
            rdbtnNewRadioButton.setVisible(false);
            rdbtnNewRadioButton_1.setVisible(false);
        }
        lblNewLabel_7 = new JLabel("eWallet ID");
        lblNewLabel_7.setFont(new Font("Tahoma", Font.PLAIN, 14));
        GridBagConstraints gbc_lblNewLabel_7 = new GridBagConstraints();
        gbc_lblNewLabel_7.anchor = GridBagConstraints.NORTH;
        gbc_lblNewLabel_7.insets = new Insets(0, 0, 5, 5);
        gbc_lblNewLabel_7.gridx = 1;
        gbc_lblNewLabel_7.gridy = 8;
```

Software Design Specification

```
        frmGofoLogin.getContentPane().add(lblNewLabel_7, gbc_lblNewLabel_
7);

        textField_6 = new JTextField();
        GridBagConstraints gbc_textField_6 = new GridBagConstraints();
        gbc_textField_6.gridwidth = 2;
        gbc_textField_6.insets = new Insets(0, 0, 5, 5);
        gbc_textField_6.fill = GridBagConstraints.HORIZONTAL;
        gbc_textField_6.gridx = 2;
        gbc_textField_6.gridy = 8;
        frmGofoLogin.getContentPane().add(textField_6, gbc_textField_6);
        textField_6.setColumns(10);

        lblNewLabel_8 = new JLabel("Balance");
        lblNewLabel_8.setFont(new Font("Tahoma", Font.PLAIN, 14));
        GridBagConstraints gbc_lblNewLabel_8 = new GridBagConstraints();
        gbc_lblNewLabel_8.anchor = GridBagConstraints.NORTH;
        gbc_lblNewLabel_8.insets = new Insets(0, 0, 5, 5);
        gbc_lblNewLabel_8.gridx = 1;
        gbc_lblNewLabel_8.gridy = 9;
        frmGofoLogin.getContentPane().add(lblNewLabel_8, gbc_lblNewLabel_
8);

        textField_7 = new JTextField();
        GridBagConstraints gbc_textField_7 = new GridBagConstraints();
        gbc_textField_7.gridwidth = 2;
        gbc_textField_7.insets = new Insets(0, 0, 5, 5);
        gbc_textField_7.fill = GridBagConstraints.HORIZONTAL;
        gbc_textField_7.gridx = 2;
        gbc_textField_7.gridy = 9;
        frmGofoLogin.getContentPane().add(textField_7, gbc_textField_7);
        textField_7.setColumns(10);

        btnNewButton = new JButton("Cancel");
        GridBagConstraints gbc_btnNewButton = new GridBagConstraints();
        gbc_btnNewButton.insets = new Insets(0, 0, 0, 5);
        gbc_btnNewButton.gridx = 1;
        gbc_btnNewButton.gridy = 10;
```

Software Design Specification

```
        frmGofoLogin.getContentPane().add(btnNewButton, gbc_btnNewButton)
;

        btnNewButton_1 = new JButton(mode==1?"Register\r\n":"Update");
        btnNewButton_1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                per=mapToPerson();
                Persons lst=new Persons();
                if(mode==1)
                {
                    VerificationCodeDlg dlg=new VerificationCodeDlg(per.g
etEmail());
                    if(!dlg.isCorrect())
                    {
                        JOptionPane.showMessageDialog(null,"You are not r
egistered");
                        return;
                    }
                    if(lst.isEmpty())per.setID(1); else per.setID(lst.siz
e()+1);
                    lst.add(per);
                }
                else {
                    lst.updateElement(per);
                }
                lst.storePersons();
                if(mode==1)
                    JOptionPane.showMessageDialog(null, "You're successfu
lly registered", "Registeration " , JOptionPane.INFORMATION_MESSAGE);
                else
                    JOptionPane.showMessageDialog(null, "Your profile is
saved", "Update" , JOptionPane.INFORMATION_MESSAGE);
                JComponent comp = (JComponent) e.getSource();
                Window win = SwingUtilities.getWindowAncestor(comp);
                win.dispose();
            }
        });
        btnNewButton .addActionListener(new ActionListener()
        {
```

Software Design Specification

```
        /**
         * to action listener with the buttons
         */

            @Override
            public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                 JComponent comp = (JComponent) e.getSource();
                  Window win = SwingUtilities.getWindowAncestor(comp);
                  win.dispose();

            }

        }
        );

        GridBagConstraints gbc_btnNewButton_1 = new GridBagConstraints();
        gbc_btnNewButton_1.insets = new Insets(0, 0, 0, 5);
        gbc_btnNewButton_1.gridx = 3;
        gbc_btnNewButton_1.gridy = 10;
        frmGofoLogin.getContentPane().add(btnNewButton_1, gbc_btnNewButto
n_1);
        frmGofoLogin.setVisible(true);
    }
    /**
     * function to set Person's informations
     * @return returning entered informations
     */
    private Person mapToPerson()
    {
        per.setName(textField.getText() + " " + textField_1.getText());
        try {
            per.setEmail(textField_2.getText());
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, e.getMessage());
            textField_2.requestFocusInWindow();
            return null;
```

Software Design Specification

```
        }
        try
        {
            per.setPassword(textField_3.getText());
            per.setPhone(textField_4.getText());
            per.setLocation(textField_5.getText());
            if(mode==1)
            {
                if(rdbtnNewRadioButton_1.isSelected())per.setRole(Person.
Role.PLAYER);
                else if(rdbtnNewRadioButton.isSelected())per.setRole(Pers
on.Role.PLAYGROUNDOWNER);
            }
            per.setEwalletID(Integer.parseInt(textField_6.getText()));
            per.setEwalletAmount(Double.parseDouble(textField_7.getText()
));

        }catch(Exception ex)
        {

        }
        return per;
    }
    /**
     * function to show your informations
     * @param per is a person who sign up
     * @return returning entered informations
     */
    public Person mapFromPerson(Person per)
    {
        try {
            String[] m=per.getName().split(" ");
            textField.setText(m[0]);
            textField_1.setText(m[1]);
        }catch(Exception ex)
        {

        }
        textField_2.setText(per.getEmail());
```

Software Design Specification

```
        textField_3.setText(per.getPassword());

        textField_4.setText(per.getPhone());
        textField_5.setText(per.getLocation());
        if(per.getRole()==Person.Role.ADMIN)
        {
            rdbtnNewRadioButton.setVisible(false);
            rdbtnNewRadioButton_1.setVisible(false);
        }else if(per.getRole()==Person.Role.PLAYER)
        {
            rdbtnNewRadioButton_1.setSelected(true);
            rdbtnNewRadioButton.setSelected(false);
        }else
        {
            rdbtnNewRadioButton_1.setSelected(false);
            rdbtnNewRadioButton.setSelected(true);
        }
        if(per.getRole()==Person.Role.PLAYER|| per.getRole()==Person.Role
.PLAYGROUNDOWNER)
        {
            textField_6.setText(String.valueOf(per.getEwalletID()));
            textField_7.setText(String.valueOf(per.getEwalletAmount()));
        }else
        {
            lblNewLabel_6.setVisible(false);
            lblNewLabel_7.setVisible(false);
            lblNewLabel_8.setVisible(false);
            textField_6.setVisible(false);
            textField_7.setVisible(false);
        }
        return per;
    }
}
```

Class: VerificationCodeDlg

```
import javax.swing.JFrame;
```

Software Design Specification

```
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Random;

import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JDialog;
/**
 * this class for VerificationCode to make person regist sucessfully
 *  Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */
@SuppressWarnings("serial")

public class VerificationCodeDlg extends JDialog {
    /**
     * JPanel to display all components on a frame
     */
    private JPanel contentPane;
    /**
     * JTextField to enter text
     */
    private JTextField textField;
    /**
     * boolean value for Verification Code
     */
    boolean correct=false;
    /**
     * function to check if Verification Code is right or not
     * @return returning boolean value
     */
```

Software Design Specification

```
    public boolean isCorrect()
    {
        return correct;
    }
    /**
     * Create the frame.
     */
    /**
     * this function to VerificationCode
     * @param email email that person entered in SignUp form
     */
    public VerificationCodeDlg(String email) {
        Random rnd=new Random();
        String genCode=String.valueOf(1000+rnd.nextInt(8999));
        MailUtil.sendEmail(email, "Gofo account verification","Your verif
ication code is:"+genCode);
        setModal(true);
        setTitle("Enter The Verification Code");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 392, 231);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNewLabel = new JLabel("Verification Code");
        lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 16));
        lblNewLabel.setBounds(45, 71, 131, 20);
        contentPane.add(lblNewLabel);

        textField = new JTextField();
        textField.setBounds(186, 73, 124, 20);
        contentPane.add(textField);
        textField.setColumns(10);

        JButton btnNewButton = new JButton("Verify");
        btnNewButton.setBounds(142, 120, 89, 23);
        btnNewButton.addActionListener(new ActionListener() {
/**
```

Software Design Specification

```
 * to action listener with the buttons
 */
            @Override
            public void actionPerformed(ActionEvent e) {
                if(genCode.equals(textField.getText()))
                        {
                        correct=true;
                        dispose();
                        }
                else
                        JOptionPane.showMessageDialog(null,"Error in veri
fication code");

            }
        }
                );
        contentPane.add(btnNewButton);
        setVisible(true);
    }

}
```

Package: persons

Class: Admin

```
package persons;

import playgrounds.*;
/**
 * this is class  Admin show add's function (what he can do) and it inher
it from class Person
 * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */
```

Software Design Specification

```
@SuppressWarnings("serial")

public class Admin extends Person{

    /**
     * function to add user
     * @param p is an object from Playground class
     */
    public void addUser(Person p) {
        // TODO - implement Admin.addUser
        throw new UnsupportedOperationException();
    }

    /**
     * function to update information of user
     * @param p is an object from Playground class
     */
    public void updateInfo(Person p) {
        // TODO - implement Admin.updateInfo
        throw new UnsupportedOperationException();
    }

    /**
     * function to approve playground
     * @param p is an object from Playground class
     */
    public void approvePlayground(Playground p) {
        // TODO - implement Admin.approvePlayground
        throw new UnsupportedOperationException();
    }

    /**
     * function to suspend  playground (playgrounds that booked , made it
s unavailable )
     * @param p is an object from Playground class
     */
    public void suspendPlayground(Playground p) {
        // TODO - implement Admin.suspendPlayground
```

Software Design Specification

```
        throw new UnsupportedOperationException();
    }

    /**
     * function to delete playground
     * @param p is an object from Playground class
     */
    public void deletePlayground(Playground p) {
        // TODO - implement Admin.deletePlayground
        throw new UnsupportedOperationException();
    }

    /**
     * function to activate playground
     * @param p is an object from Playground class
     */
    public void activatePlayground(Playground p) {
        // TODO - implement Admin.activatePlayground
        throw new UnsupportedOperationException();
    }

}
```

Package: persons

Class: Adminstrator

```
/**
 *
 */
package persons;

/**
 * @author noura
 *
 */
public class Adminstrator {
```

```
}
```

Package: persons

Class: Person

```
package persons;

import java.io.Serializable;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 *
 * this class to make Sign in  , sign up ,create my profile
 * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
 * @version 1.0
 *
 */


@SuppressWarnings("serial")

public class Person implements Serializable {
/**
 * enum for role
 */
    public enum Role {ADMIN,PLAYER,PLAYGROUNDOWNER};
    /**
     * String value for the first attribute
     */
    private String name;
    /**
     * String value for the second attribute
     */
```

Software Design Specification

```
    private String password;
    /**
     * Int value for the third attribute
     */
    private int ID;
    /**
     * String value for the forth attribute
     */
    private String email;
    /**
     * String value for the fifth attribute
     */
    private String phone;
    /**
     * String value for the six attribute
     */
    private String location;
    /**
     * enum value for role attribute
     */
    private Role role;
    /**
     * Int value for the eight attribute
     */
    private int ewalletID;
    /**
     * Double value for the ninth attribute
     */
    private double ewalletAmount;
/**
 * function to create my profile
 */
    public void createMyProfile() {
        // TODO - implement Person.createMyProfile
        throw new UnsupportedOperationException();
    }
/**
 * function to log in
```

Software Design Specification

```
 */
    public void login() {
        // TODO - implement Person.login
        throw new UnsupportedOperationException();
    }
/**
 * function to sign up
 */
    public void signUp() {
        // TODO - implement Person.signUp
        throw new UnsupportedOperationException();
    }
/**
 * This method to get name value.
 * @return returning string value (user's name)
 */
    public String getName() {
        return name;
    }
/**
 * This method to set certain value to name (attribute).
 * @param name is setter value to set in name.
 */
    public void setName(String name) {
        this.name = name;
    }
/**
 * This method to get password value.
 * @return returning string value (user's password)
 */
    public String getPassword() {
        return password;
    }
/**
 * This method to set certain value to password (attribute).
 * @param password is setter value to set in password
 */
    public void setPassword(String password) {
```

Software Design Specification

```
        this.password = password;
    }
/**
 * This method to get ID value.
 * @return returning int value (user's ID)
 *
 */
    public int getID() {
        return ID;
    }
/**
 *  This method to set certain value to iD (attribute).
 *  @param iD is setter value to set in iD
 *
 */
    public void setID(int iD) {
        ID = iD;
    }
/**
 * This method to get email value.
 * @return returning string value (user's email)
 *
 */
    public String getEmail() {
        return email;
    }
/**
 * function to set certain value to email (attribute) and check email is
valid or not
 * @param email is setter value to set in email
 * @throws Exception
 */
    public void setEmail (String email) throws Exception {
        String ch="^\\w+([\\.-]?\\w+)*@\\w+([\\.-
]?\\w+)*(\\.\\w{2,3})+$";
        Pattern pattern = Pattern.compile(ch,Pattern.CASE_INSENSITIVE);
        Matcher matcher = pattern.matcher(email);
        if(matcher.matches())
```

Software Design Specification

```
        this.email = email;
        else
        {
            this.email = email;
            throw new Exception("Email is not correct");
        }
    }
/**
 * This method to get phone value.
 * @return returning string value (user's phone)
 *
 */
    public String getPhone() {
        return phone;
    }
/**
 *  This method to set certain value to phone (attribute).
 *
 * @param phone is setter value to set in phone
 */
    public void setPhone(String phone) {
        this.phone = phone;
    }
/**
 * This method to get location value.
 * @return returning string value (user's location)
 */
    public String getLocation() {
        return location;
    }
/**
 * This method to set certain value to location (attribute).
 * @param location is setter value to set in location
 */
    public void setLocation(String location) {
        this.location = location;
    }
/**
```

Software Design Specification

```
 * This method to get role value.
 * @return returning value from enum
 */
    public Role getRole() {
        return role;
    }
/**
 * This method to set certain value to role (attribute).
 * @param role is setter value to set in role
 */
    public void setRole(Role role) {
        this.role = role;
    }
/**
 * This method to get ewalletID value.
 * @return returning int value (user's ewalletID)
 */
    public int getEwalletID() {
        return ewalletID;
    }
/**
 * This method to set certain value to ewalletID (attribute).
 * @param ewalletID is setter value to set in ewalletID
 */
    public void setEwalletID(int ewalletID) {
        this.ewalletID = ewalletID;
    }
/**
 * This method to get Ewallet Amount value.
 * @return returning double value (user's Ewallet Amount)
 */
    public double getEwalletAmount() {
        return ewalletAmount;
    }
/**
 * This method to set certain value to ewalletAmount (attribute).
 * @param ewalletAmount is setter value to set in ewalletAmount
 */
```

Software Design Specification

```
    public void setEwalletAmount(double ewalletAmount) {
        this.ewalletAmount = ewalletAmount;
    }
    /**
     * this method to add to user's ewallet
     * @param amount to set amount  in ewalletAmount after adding
     */
    public void addToWallet(double amount)
    {
        ewalletAmount+=amount;
    }
    /**
     * this method to remove from user's ewallet
     * @param amount to set amount in ewalletAmount after removing
     */
    public void removeFromWallet(double amount)
    {
        ewalletAmount-=amount;
    }
}
```

Package: persons

Class: Persons

```
package persons;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;


/**
 * this class to add persons in array list and store it in file
 * Date:10-jun-2020
 * @author noura,eman,alaa,neimat
```

Software Design Specification

```
 * @version 1.0
 *
 */
public class Persons {
    /**
     * array list to add persons
     */
    private ArrayList<Person> lst=new ArrayList<Person>();
    public Persons()
    {
        loadPersons();
    }
    /**
     * function to read persons from file
     */

    public void loadPersons()
    {
        try {
            FileInputStream fis = new FileInputStream("Persons.bin");
            ObjectInputStream ois = new ObjectInputStream(fis);
            lst=(ArrayList<Person>)ois.readObject();
            ois.close();
            fis.close();
            } catch (IOException e1) {
            lst=new ArrayList<Person>();
        } catch (ClassNotFoundException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
    /**
     * function to store persons in file
     */
    public void storePersons()
    {
          try{
              FileOutputStream fos= new FileOutputStream("Persons.bin");
```

Software Design Specification

```
            ObjectOutputStream oos= new ObjectOutputStream(fos);
            oos.writeObject(lst);
            oos.close();
            fos.close();
          }catch(IOException ioe){
              ioe.printStackTrace();
           }
    }
    /**
     * function to get all persons i added to array list
     * @return  1st (person with his information)
     */
    public ArrayList<Person> getList()
    {
        return lst;
    }
    /**
     * function to delete all elements from array list
     */
    public void deletearr()
    {
        lst.removeAll(lst);
    }
    /**
     * function to check if array list is empty or not
     * @return 1st ( removed person from array list)
     */
    public boolean isEmpty()
    {
        return lst.isEmpty();
    }
    /**
     * function of arraylist's size
     * @return size of array list
     */
    public int size()
    {
        return lst.size();
```

Software Design Specification

```
    }
    /**
     * function to add person in array list
     * @param per to add person in array list
     */
    public void add(Person per)
    {
        lst.add(per);
    }
    /**
     * function to find person by email and check his email is valid or n
ot
     * @param email to enter email
     * @param pass to enter password
     * @return a person of entered his email and pass
     */
    public Person findPersonByEmail(String email, String pass)
    {
        String regex="^[\\w-_\\.+]*[\\w-
_\\.]\\@([\\w]+\\.)+[\\w]+[\\w]$";

        for(Person per:lst)
        {
            //JOptionPane.showMessageDialog(null, "ayhaga");
            if(per.getEmail().equals(email) && per.getPassword().equals(p
ass))
            {
                if( email.matches(regex))
                {
                    return per;
                }

            }

        }
        return null;
    }
```

Software Design Specification

```
    /**
     * function to find person by enter ID
     * @param ID to eneter ID
     * @return a person of entered ID
     */
    public Person findPersonByID(int ID)
    {
        for(Person per:lst)
        {
            if(per.getID()==ID)
            {
                return per;
            }
        }
        return null;
    }
    /**
     * function to get person from array list
     * @param i for person
     * @return a person from array list
     */
    public Person get(int i) {
        // TODO Auto-generated method stub
        return lst.get(i);
    }
    /**
     * function to update person's information in array list
     * @param per
     */
    public void updateElement(Person per) {
        for(int i=0;i<lst.size();i++)
            if(per.getID()==lst.get(i).getID())
            {
                lst.set(i, per);
                break;
            }
    }
}
```

Software Design Specification

Package: persons

Class: Player

```
package persons;

import java.sql.Time;
import java.util.Date;

import playgrounds.BookingSlot;
import playgrounds.Playground;

@SuppressWarnings("serial")
/**
 * this class to implement all methods that player want to do and it inhe
rit from class person
 * Date : 10-jun-2020
 * @author noura
 * @version 1.0
 *
 */
public class Player extends Person {
/**
 * function to update profile
 */
    public void updateProfile() {
        // TODO - implement Player.updateProfile
        throw new UnsupportedOperationException();
    }
/**
 * function to show playgrounds of playground owner
 */
    public void showPlaygrounds() {
        // TODO - implement Player.showPlaygrounds
        throw new UnsupportedOperationException();
```

Software Design Specification

```
    }

    /**
     * function to  book a playground slot
     * @param s for set a booking slot
     */
    public void bookAPlaygroundSlot(BookingSlot s) {
        // TODO - implement Player.bookAPlaygroundSlot
        throw new UnsupportedOperationException();
    }

    /**
     * function to calculate price of booked playground
     * @param p booked playground
     * @param d date of booked playground
     * @param t time of booked playground
     */
    public double calculatePrice(Playground p, Date d, Time t) {
        // TODO - implement Player.calculatePrice
        throw new UnsupportedOperationException();
    }
/**
 * function to cancel booking
 */
    public void cancleBooking() {
        // TODO - implement Player.cancleBooking
        throw new UnsupportedOperationException();
    }
/**
 * function to invite team
 */
    public void inviteTeam() {
        // TODO - implement Player.inviteTeam
        throw new UnsupportedOperationException();
    }
/**
 * function to create team
 */
```

Software Design Specification

```
    public void createTeam() {
        // TODO - implement Player.createTeam
        throw new UnsupportedOperationException();
    }
/**
 * function to view playing slots
 */
    public void viewPlayingSlots() {
        // TODO - implement Player.viewPlayingSlots
        throw new UnsupportedOperationException();
    }

}
```

Package: persons

Class: PlaygroundOwner

```
package persons;

import playgrounds.*;

@SuppressWarnings("serial")
/**
 * this class implement all methods that playground owner want to do and
it inherit from class person
 * Date : 10-jun-2020
 * @author eman,noura,alaa,neimat
 * @version 1.0
 *
 */
public class PlaygroundOwner extends Person {

    /**
     * function to add playground
```

Software Design Specification

```
     * @param p is object from class Playground to add it to the array li
st
     */
    public void addPlayground(Playground p) {
        // TODO - implement PlaygroundOwner.addPlayground
        throw new UnsupportedOperationException();
    }

    /**
     * function to update playground info
     * @param p is object from class Playground to update it to the array
 list
     */
    public void updatePlaygroundInfo(Playground p) {
        // TODO - implement PlaygroundOwner.updatePlaygroundInfo
        throw new UnsupportedOperationException();
    }
/**
 * function to check ewallet
 * @return a boolean value express if ewallet is full or empty
 */
    public boolean checkEWallet() {
        // TODO - implement PlaygroundOwner.checkEWallet
        throw new UnsupportedOperationException();
    }

}
```

Package: persons

Class: Team

```
package persons;

import java.io.Serializable;
import java.util.Vector;
```

Software Design Specification

```
@SuppressWarnings("serial")
/**
 * this class for team
 * Date:10-jun-2020
 * @author noura, eman,neimat,alaa
 * @version 1.0
 */
public class Team implements Serializable {
/**
 * Int value for the first attribute
 */
    private int ID;
    /**
     * String value for the second attribute
     */
    private String name;
    /**
     * vector to store team's member in it
     */
    private Vector<Integer> members=new Vector<Integer>();
/**
 * * This method to get name value.
 * @return returning string value (team's member name)
 *
 */
        public String getName() {
        return this.name;
    }

    /**
     * This method to set certain value to name (attribute).
     * @param name is setter value to set in name.

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

Software Design Specification

```
    /**
     * this method to add player in team
     * @param pID is  player ID i want to add in team
     */
    public void addPlayer(int pID) {
        members.add(pID);
    }

    /**
     * this method to remove player from team
     * @param p is player i want to remove from team
     */
    public void removePlayer(Player p) {
        // TODO - implement Team.removePlayer
        throw new UnsupportedOperationException();
    }
/**
 * constructor for class team
 */
    public Team() {

    }
/**
 * function to get team's member from vector
 * @return returning team's members
 */
    public Vector<Integer> getMembers() {
        return members;
    }
/**
 * This method to set certain value to members (attribute).
 * @param members is setter value to set in members
 */
    public void setMembers(Vector<Integer> members) {
        this.members = members;
    }
/**
```

Software Design Specification

```
 * This method to get ID value.
 * @return returning int value (member's ID)
 *
 */
    public int getID() {
        return ID;
    }
/**
 *  This method to set certain value to iD (attribute).
 * @param iD is setter value to set in iD
 */
    public void setID(int iD) {
        ID = iD;
    }
    /**
     * function to check if player is member in team or not
     * @param id is the member's id
     * @return boolean value express  if player is member in team or not
     */
    public boolean isAMember(int id)
    {
        for(int p:members)
        {
            if(p==id)return true;
        }
        return false;
    }
}
```

Package: persons

Class: Teams

```
package persons;

import java.io.FileInputStream;
```

Software Design Specification

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
/**
 * this class for team
 * Date:10-jun-2020
 * @author noura,eman,neimat,alaa
 * @version 1.0
 */
public class Teams {
    /**
     * array list of team members
     */
    private ArrayList<Team> lst=new ArrayList<Team>();
    /**
     * default constructor
     */
    public Teams()
    {
        loadTeams();
    }
    /**
     * function to read from file
     */
    public void loadTeams()
    {
        try {
            FileInputStream fis = new FileInputStream("Teams.bin");
            ObjectInputStream ois = new ObjectInputStream(fis);
            lst=(ArrayList<Team>)ois.readObject();
            ois.close();
            fis.close();
            } catch (IOException e1) {
            lst=new ArrayList<Team>();
        } catch (ClassNotFoundException e1) {
            // TODO Auto-generated catch block
```

Software Design Specification

```
            e1.printStackTrace();
        }
    }
    /**
     * function to store team's members in file
     */
    public void storeTeams()
    {
          try{
              FileOutputStream fos= new FileOutputStream("Teams.bin");
              ObjectOutputStream oos= new ObjectOutputStream(fos);
              oos.writeObject(lst);
              oos.close();
              fos.close();
            }catch(IOException ioe){
                ioe.printStackTrace();
             }
    }
    /**
     * this function to get team's members from array list
     * @return lst (member index with its information)
     */
    public ArrayList<Team> getList()
    {
        return lst;
    }
    /**
     * function to check if array list is empty or not
     * @return lst ( removed member from array list)
     *
     */
    public boolean isEmpty()
    {
        return lst.isEmpty();
    }
    /**
     * this function of size of array list
     * @return lst.size (size of members in array list )
```

Software Design Specification

```
     */
    public int size()
    {
        return lst.size();
    }
    /**
     * function to add team
     * @param team is object from class Team  to add team in array list
     */
    public void add(Team team)
    {
        lst.add(team);
    }
    /**
     * function to get team
     * @param index is the index of  member in team
     * @return lst (team member i want)
     */
    public Team get(int index)
    {
        return lst.get(index);
    }
    /**
     * function to find team by id
     * @param ID of the team
     * @return tm (team i entered its ID)
     */
    public Team findTeamByID(int ID)
    {
        for(Team tm:lst)
        {
            if(tm.getID()==ID)
            {
                return tm;
            }
        }
        return null;
    }
```

```
}
```

Package: playgrounds

Class: AvailabilityStatus

```
package playgrounds;
/**
 * this is class for Availability Status of playground(ACTIVE,SUSPENDED,D
ELETED)
 * Date:10-jun-2020
 * @author noura,eman,neimat,alaa
 * @version 1.0
 */
public enum AvailabilityStatus {
    ACTIVE,
    SUSPENDED,
    DELETED
}
```

Package: playgrounds

Class: BookingSchedule

```
package playgrounds;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
/**
 * this class for book playground
 * Date:10-jun-2020
 * @author noura,eman,neimat,alaa
```

Software Design Specification

```
 * @version 1.0
 */
public class BookingSchedule {
/**
 * array list for booking slot of playground
 */
    private ArrayList<BookingSlot> slots;
/**
 * function to read booking slot from file
 */
    public void loadSchedule()
    {
        try {
            FileInputStream fis = new FileInputStream("Schedule.bin");
            ObjectInputStream ois = new ObjectInputStream(fis);
            slots=(ArrayList<BookingSlot>)ois.readObject();
            ois.close();
            fis.close();
            } catch (IOException e1) {
            slots=new ArrayList<BookingSlot>();
        } catch (Exception e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
    /**
     * function to store booking slot in file
     */
    public void storeSchedule()
    {
          try{
              FileOutputStream fos= new FileOutputStream("Schedule.bin");
              ObjectOutputStream oos= new ObjectOutputStream(fos);
              oos.writeObject(slots);
              oos.close();
              fos.close();
            }catch(IOException ioe){
                ioe.printStackTrace();
```

Software Design Specification

```
            }
    }
/**
 * function to add slot of playground
 * @param s is an object of class BookingSlot to add in array list
 */
    public void addSlot(BookingSlot s) {
        slots.add(s);
    }

    /**
     *
     * function to remove slot of playground
     * @param bs is an object of class BookingSlot to remove in array lis
t
     */
    public void removeSlot(BookingSlot bs) {
        // TODO - implement BookingSchedule.removeSlot
        throw new UnsupportedOperationException();
    }

    /**
     * function to change slot status
     * @param bs is an object from BookingSlot class
     * @param s boolean value to express about if slots status is changed
 or not
     */
    public void changeSlotStatus(BookingSlot bs, boolean s) {
        // TODO - implement BookingSchedule.changeSlotStatus
        throw new UnsupportedOperationException();
    }
/**
 * default constructor
 */
    public BookingSchedule() {
        loadSchedule();
    }
/**
```

Software Design Specification

```
 * function to get all slots i added to array list
 * @return slots (slots with its information)
 */
    public ArrayList<BookingSlot> getSlots() {
        return slots;
    }
/**
 * function to set slots of playground in array list
 * @param slots is an object from array list to set slots in it
 */
    public void setSlots(ArrayList<BookingSlot> slots) {
        this.slots = slots;
    }
}
```

Package: playgrounds

Class: BookingSlot

```
package playgrounds;

import java.io.Serializable;
import java.time.LocalTime;
import java.util.Date;

@SuppressWarnings("serial")
/**
 * this class for booking slot
 * Date:10-jun-2020
 * @author noura,eman,neimat,alaa
 * @version: 1.0
 */
public class BookingSlot implements Serializable{
/**
 *  Date value for the first attribute
 */
```

Software Design Specification

```
    private Date slotDate;
    /**
     * LocalTime value for the second attribute
     */
    private LocalTime startTime;
    /**
     * LocalTime value for the third attribute
     */
    private LocalTime endTime;
    /**
     * boolean value for the forth attribute
     */
    private boolean status;
    /**
     * int value for the fifth attribute
     */
    private int playerID;
    /**
     * int value for the first attribute
     */
    private int playgroundID;
/**
 *  This method to get slotDate value.
 * @return returning date value (slotDate)
 */
    public Date getSlotDate() {
        return this.slotDate;
    }

    /**
     *  This method to set certain value to slotDate (attribute).
     * @param slotDate is setter value to set in slotDate.
     */
    public void setSlotDate(Date slotDate) {
        this.slotDate = slotDate;
    }
/**
 *  This method to get startTime value.
```

Software Design Specification

```
 * @return returning LocalTime  value (booking startTime)
 */
    public LocalTime getStartTime() {
        return this.startTime;
    }

    /**
     * This method to set certain value to startTime  (attribute).
     * @param startTime is setter value to set in startTime.
     */
    public void setStartTime(LocalTime startTime) {
        this.startTime = startTime;
    }
/**
 *  This method to get endTime value.
 * @return returning LocalTime value (booking endTime)
 */
    public LocalTime getEndTime() {
        return this.endTime;
    }

    /**
     * This method to set certain value to endTime (attribute).
     * @param endTime is setter value to set in endTime.
     */
    public void setEndTime(LocalTime endTime) {
        this.endTime = endTime;
    }
/**
 *  This method to get status value.
 * @return returning boolean value (playground's status)
 */
    public boolean getStatus() {
        return this.status;
    }

    /**
     * This method to set certain value to status (attribute).
```

Software Design Specification

```
     * @param status is setter value to set in status.
     */
    public void setStatus(boolean status) {
        this.status = status;
    }
/**
 *  This method to get playerID value.
 * @return returning int value (playerID)
 */
    public int getPlayerID() {
        return playerID;
    }
/**
 * This method to set certain value to playerID (attribute).
 * @param playerID is setter value to set in playerID.
 */
    public void setPlayerID(int playerID) {
        this.playerID = playerID;
    }
/**
 *  This method to get playgroundID value.
 * @return returning int value (playground ID)
 */
    public int getPlaygroundID() {
        return playgroundID;
    }
/**
 * This method to set certain value to playgroundID (attribute).
 * @param playgroundID is setter value to set in playgroundID.
 */
    public void setPlaygroundID(int playgroundID) {
        this.playgroundID = playgroundID;
    }
}
```

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

Software Design Specification

Package: playgrounds

Class: Playground

```
package playgrounds;

import java.io.Serializable;

@SuppressWarnings("serial")
/**
 * this is playground class
 * Date:10-jun-2020
 * @author noura,eman,neimat,alaa
 * @version 1.0
 *
 */
public class Playground implements Serializable {
/**
 * String  value for the first attribute
 */
    private String playgroundName;
    /**
     * int value for the second attribute
     */
    private int playgroundID;
    /**
     * String value for the third attribute
     */
    private String playgroundLocation;
    /**
     * Double value for the fourth attribute
     */
    private double playgroundSize;
    /**
     * float value for the fifth attribute
     */
    private float playgroundHours;
    /**
```

Software Design Specification

```
     * object from enum
     */
    private AvailabilityStatus playgroundAvailbility;
    /**
     * Double value for the sixth attribute
     */
    private double playgroundPrice;
    /**
     * String value for the seventh attribute
     */
    private String playgroundComplain;
    /**
     * float value for the eigth attribute
     */
    private float playgroundCanclePeriod;
    /**
     * int value for the ninth attribute
     */
    private int playgroundOwnerID;
    /**
     * This method to get Playground name value.
     * @return returning string value (Playground's name)
     */
    public String getPlaygroundName() {
        return playgroundName;
    }
    /**
     * This method to set certain value to Playground name (attribute).
     * @param playgroundName is setter value to set in playgroundName.
     */
    public void setPlaygroundName(String playgroundName) {
        this.playgroundName = playgroundName;
    }
    /**
     * This method to get PlaygroundID value.
     * @return returning int value (Playground ID)
     */
    public int getPlaygroundID() {
```

Software Design Specification

```
        return playgroundID;
    }
    /**
     * This method to set certain value to PlaygroundID (attribute).
     * @param playgroundID is setter value to set PlaygroundID
     */
    public void setPlaygroundID(int playgroundID) {
        this.playgroundID = playgroundID;
    }
    /**
     * This method to get PlaygroundLocation value.
     * @return returning string value (PlaygroundLocation)
     */
    public String getPlaygroundLocation() {
        return playgroundLocation;
    }
    /**
     * This method to set certain value to PlaygroundLocation (attribute)
.
     * @param playgroundLocation is setter value to set in PlaygroundLoca
tion.
     */
    public void setPlaygroundLocation(String playgroundLocation) {
        this.playgroundLocation = playgroundLocation;
    }
    /**
     * This method to get PlaygroundSize value.
     * @return returning double value (PlaygroundSize)
     */
    public double getPlaygroundSize() {
        return playgroundSize;
    }
    /**
     * This method to set certain value to playgroundSize (attribute).
     * @param playgroundSize is setter value to set in playgroundSize.
     */
    public void setPlaygroundSize(double playgroundSize) {
        this.playgroundSize = playgroundSize;
```

Software Design Specification

```
    }
    /**
     * This method to get PlaygroundHours value.
     * @return returning float value (PlaygroundHours)
     */
    public float getPlaygroundHours() {
        return playgroundHours;
    }
    /**
     * This method to set certain value to  playgroundHours (attribute).
     * @param playgroundHours  is setter value to set in playgroundHours.
     */
    public void setPlaygroundHours(float playgroundHours) {
        this.playgroundHours = playgroundHours;
    }
    /**
     * This method to get PlaygroundAvailbility  from AvailabilityStatus
class.
     * @return returning string  value from enum (PlaygroundAvailbility)
     */
    public AvailabilityStatus getPlaygroundAvailbility() {
        return playgroundAvailbility;
    }
    /**
     * This method to set certain value to PlaygroundAvailbility (attribu
te).
     * @param playgroundAvailbility is setter value to set in PlaygroundA
vailbility.
     */
    public void setPlaygroundAvailbility(AvailabilityStatus playgroundAva
ilbility) {
        this.playgroundAvailbility = playgroundAvailbility;
    }
    /**
     * This method to get playgroundPrice value.
     * @return returning double value (playgroundPrice)
     */
    public double getPlaygroundPrice() {
```

Software Design Specification

```
        return playgroundPrice;
    }
    /**
     * This method to set certain value to playgroundPrice (attribute).
     * @param playgroundPrice is setter value to set in playgroundPrice.
     */
    public void setPlaygroundPrice(double playgroundPrice) {
        this.playgroundPrice = playgroundPrice;
    }
    /**
     * This method to get playgroundComplain value.
     * @return returning string value (playgroundComplain)
     */
    public String getPlaygroundComplain() {
        return playgroundComplain;
    }
    /**
     * This method to set certain value to playgroundComplain (attribute)
.
     * @param playgroundComplain is setter value to set in playgroundComp
lain.
     */
    public void setPlaygroundComplain(String playgroundComplain) {
        this.playgroundComplain = playgroundComplain;
    }
    /**
     * This method to get playgroundCanclePeriod value.
     * @return returning string value (playgroundCanclePeriod)
     */
    public float getPlaygroundCanclePeriod() {
        return playgroundCanclePeriod;
    }
    /**
     * This method to set certain value to playgroundCanclePeriod (attrib
ute).
     * @param playgroundCanclePeriod is setter value to set in playground
CanclePeriod.
     */
```

Software Design Specification

```
    public void setPlaygroundCanclePeriod(float playgroundCanclePeriod) {
        this.playgroundCanclePeriod = playgroundCanclePeriod;
    }
    /**
     * This method to get playgroundOwnerID value.
     * @return returning int value (playgroundOwnerID)
     */
    public int getPlaygroundOwnerID() {
        return playgroundOwnerID;
    }
    /**
     * This method to set certain value to playgroundOwnerID (attribute).
     * @param playgroundOwnerID  is setter value to set in playgroundOwne
rID.
     */
    public void setPlaygroundOwnerID(int playgroundOwnerID) {
        this.playgroundOwnerID = playgroundOwnerID;
    }

}
```

Package: playgrounds

Class: Playgrounds

```
package playgrounds;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
/**
 * this class for playgrounds
 * Date:10-jun-2020
```

Software Design Specification

```
 * @author noura,eman,neimat,alaa
 * @version 1.0
 *
 */
public class Playgrounds {
    /**
     * array list to add Playground
     */
    private ArrayList<Playground> lst=new ArrayList<Playground>();
    /**
     * default constructor
     */
    public Playgrounds()
    {
        loadPlaygrounds();
    }
    /**
     * function to read playgrounds from file
     */
    public void loadPlaygrounds()
    {
        try {
            FileInputStream fis = new FileInputStream("Playgrounds.bin");
            ObjectInputStream ois = new ObjectInputStream(fis);
            lst=(ArrayList<Playground>)ois.readObject();
            ois.close();
            fis.close();
            } catch (IOException e1) {
            lst=new ArrayList<Playground>();
        } catch (Exception e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
    /**
     * function to store playgrounds in file
     */
    public void storePlaygrounds()
```

Software Design Specification

```
    {
          try{
              FileOutputStream fos= new FileOutputStream("Playgrounds.bin
");
              ObjectOutputStream oos= new ObjectOutputStream(fos);
              oos.writeObject(lst);
              oos.close();
              fos.close();
            }catch(IOException ioe){
                ioe.printStackTrace();
             }
    }
    /**
     * function to get all playgrounds i added to array list
     * @return  1st (playground with its information)
     *
     */
    public ArrayList<Playground> getList()
    {
        return lst;
    }
    /**
     * function to check if array list is empty or not
     * @return 1st ( removed playground from array list)
     */

    public boolean isEmpty()
    {
        return lst.isEmpty();
    }
    /**
     * function of arraylist's size
     * @return size of array list
     *
     */
    public int size()
    {
        return lst.size();
```

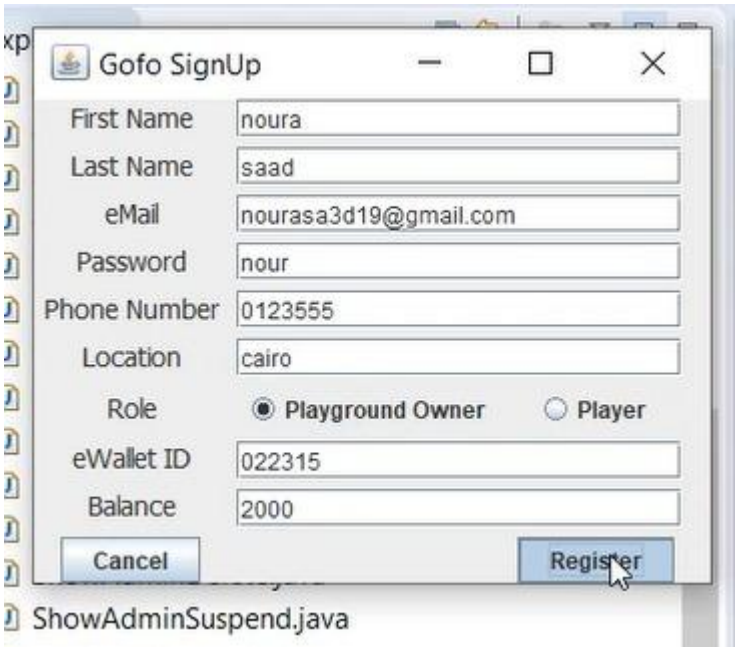
Software Design Specification

```
    }
    /**
     * function to add playground in array list
     * @param pgnd is an object from playground class to add playgrounds
in array list
     */
    public void add(Playground pgnd)
    {
        lst.add(pgnd);
    }
    /**
     * function to find Playground By entered its ID
     * @param ID is playground's ID
     * @return a playground of entered ID
     */
    public int findPlaygroundByID(int ID)
    {
        int i=0;
        for(Playground pgnd:lst)
        {
            if(pgnd.getPlaygroundID()==ID)
            {
                return i;
            }else i++;
        }
        return i;
    }
    /**
     * function to set playground
     * @param index is the index of playground in array list
     * @param pgnd is an object from playground class
     */
    public void set(int index, Playground pgnd)
    {
        lst.set(index, pgnd);
    }
    /**
     * function to get playground
```

Software Design Specification

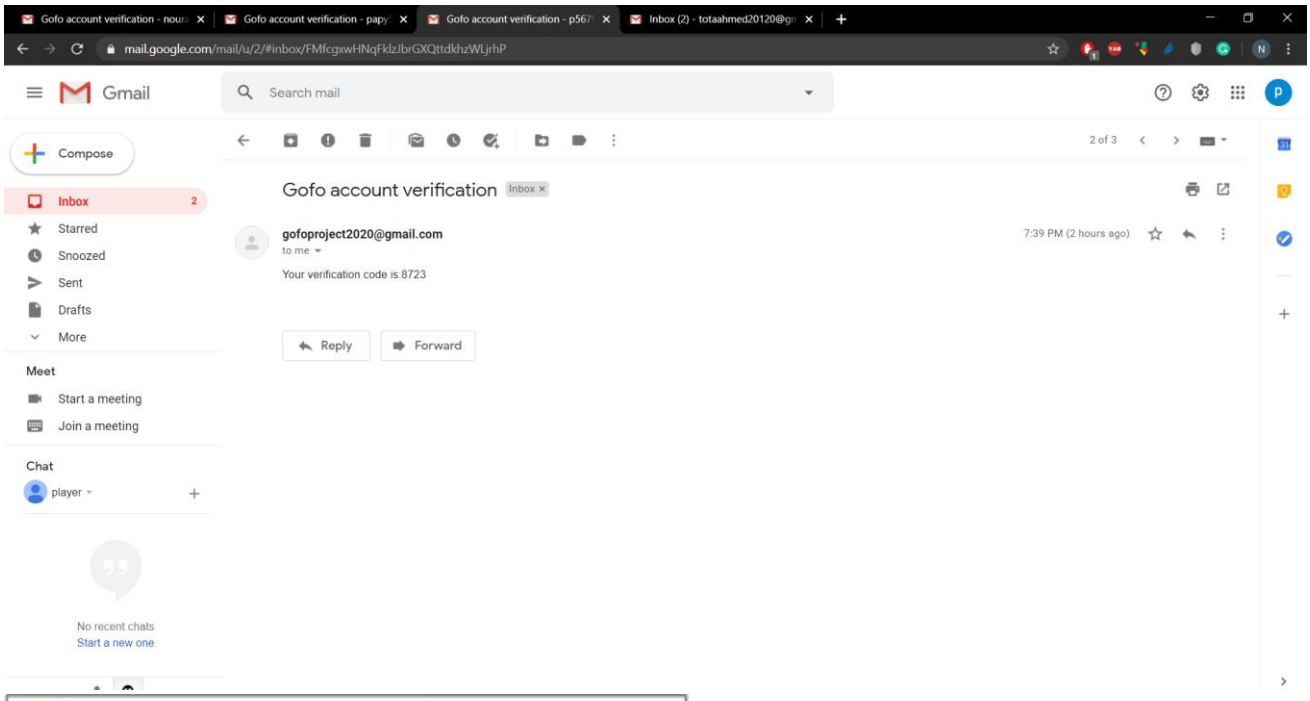
```
     * @param index is the index of playground in array list
     * @return returning playground with its index
     */
    public Playground get(int index)
    {
        return lst.get(index);
    }
}
```

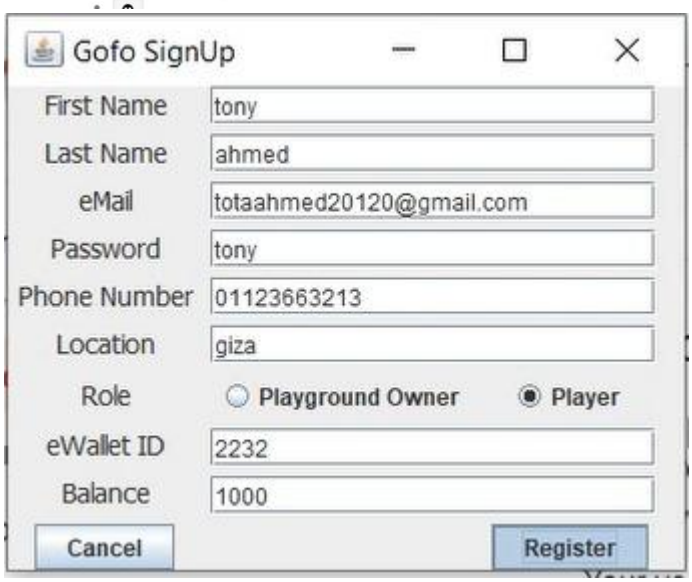
- **The output of the program:**
  For the new user



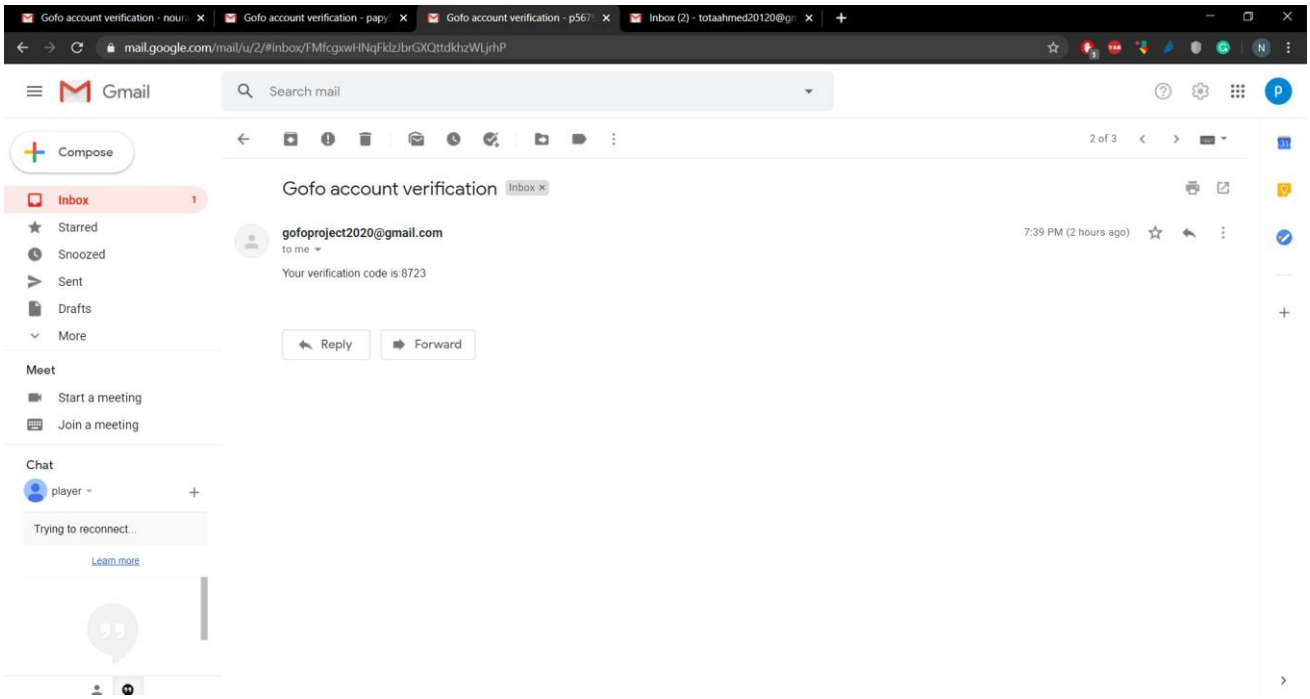
CS251: Phase 2 – <Team Name>
Project: <Project Name>

Software Design Specification

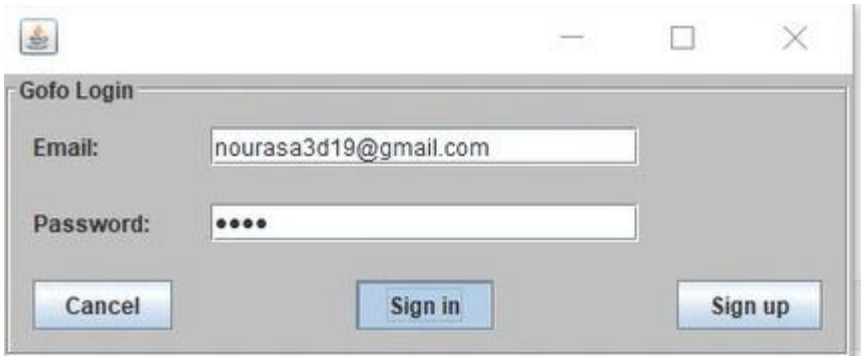




Software Design Specification

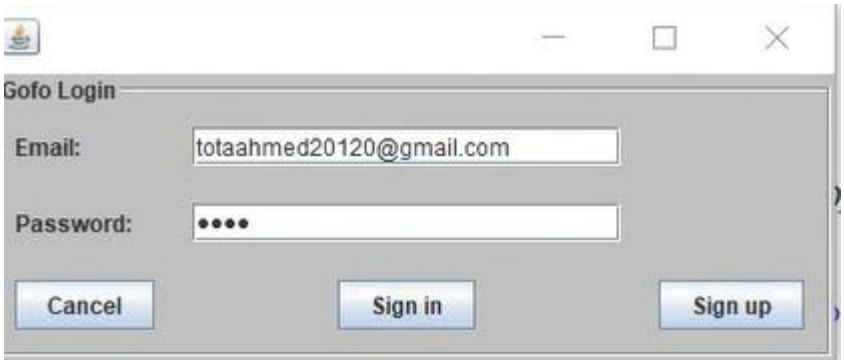


For sign in as playground owner

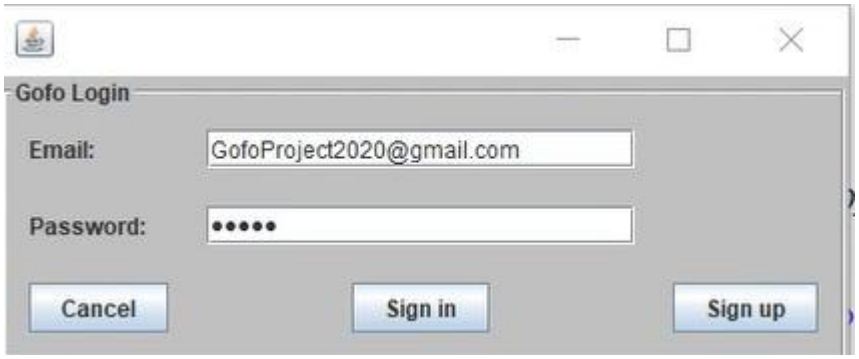


Software Design Specification

player

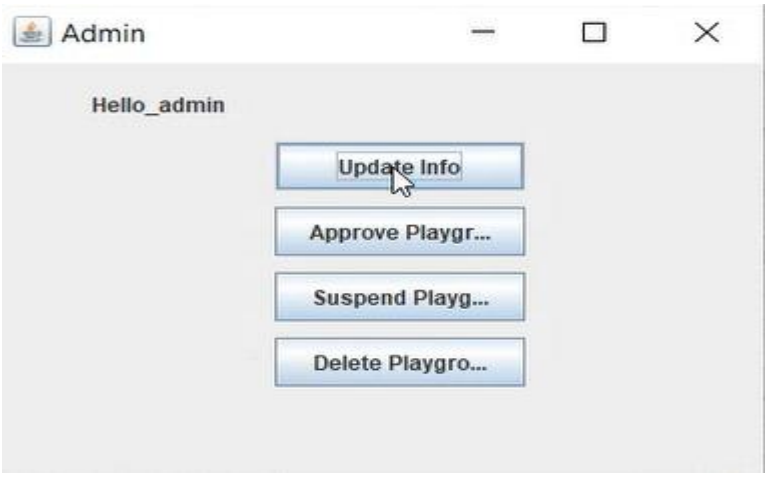


admin

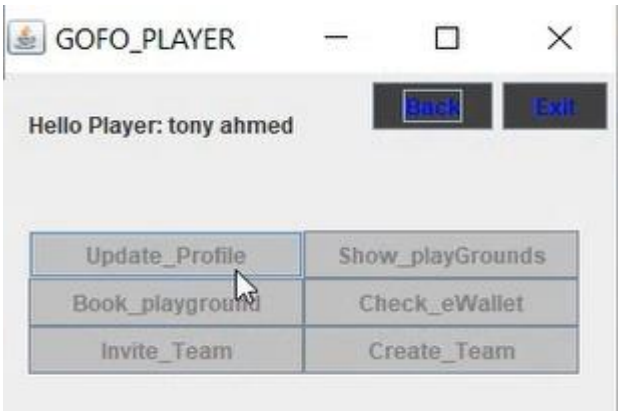


For admin options

Software Design Specification

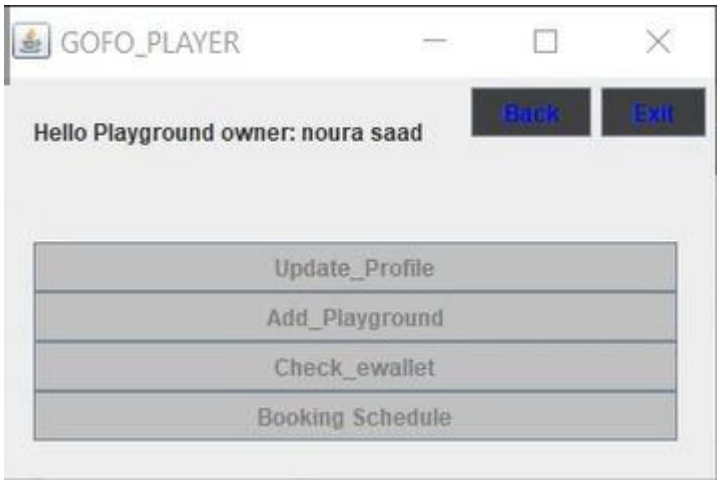


For player options

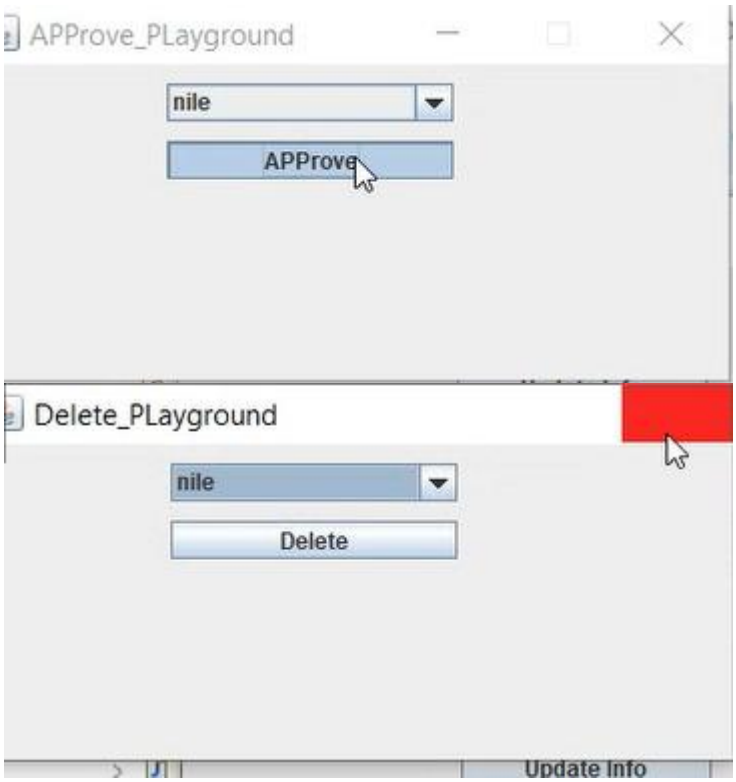


Software Design Specification

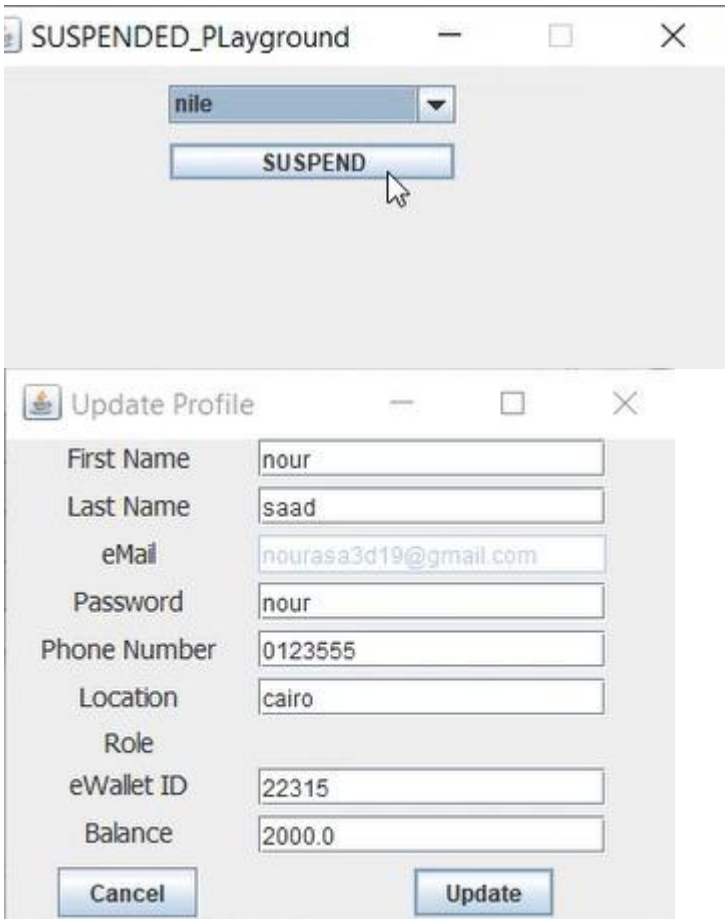
for playgroundowner



Admin options



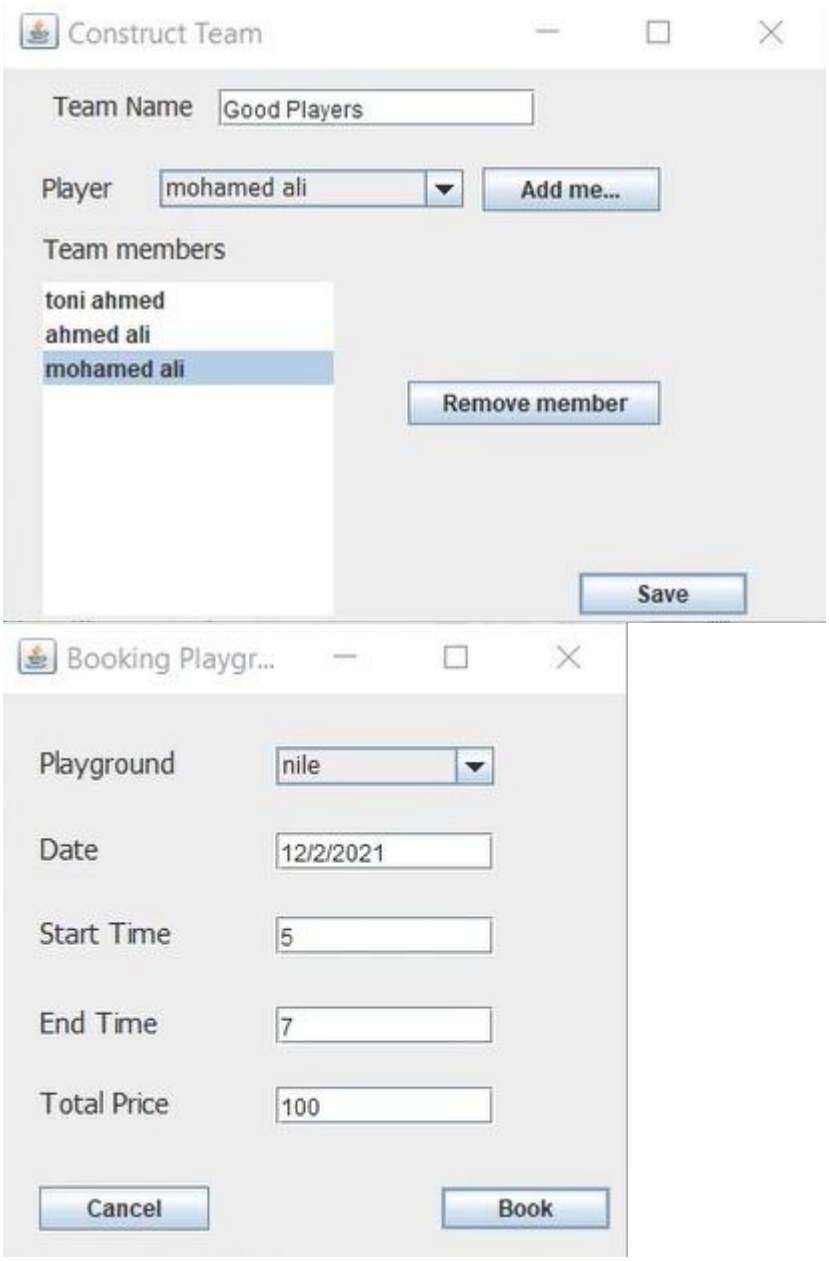
Software Design Specification



For player options

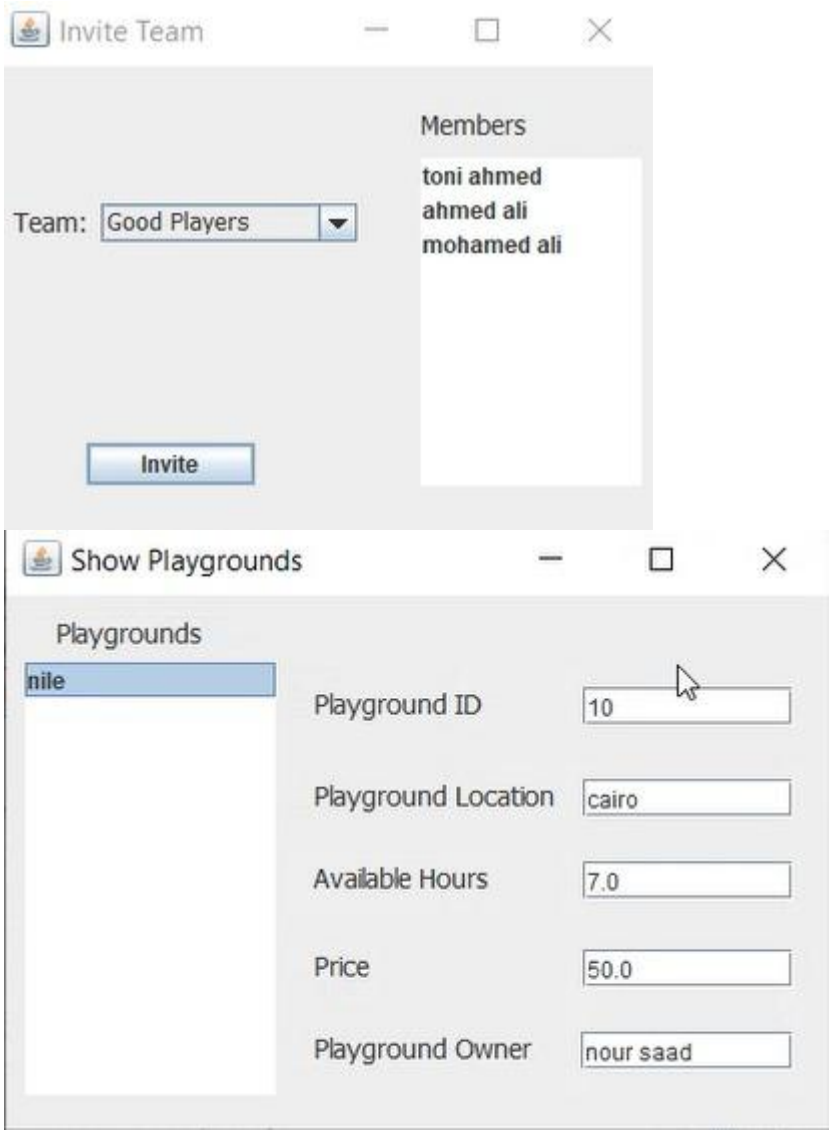
CS251: Phase 2 – <Team Name>
Project: <Project Name>

Software Design Specification

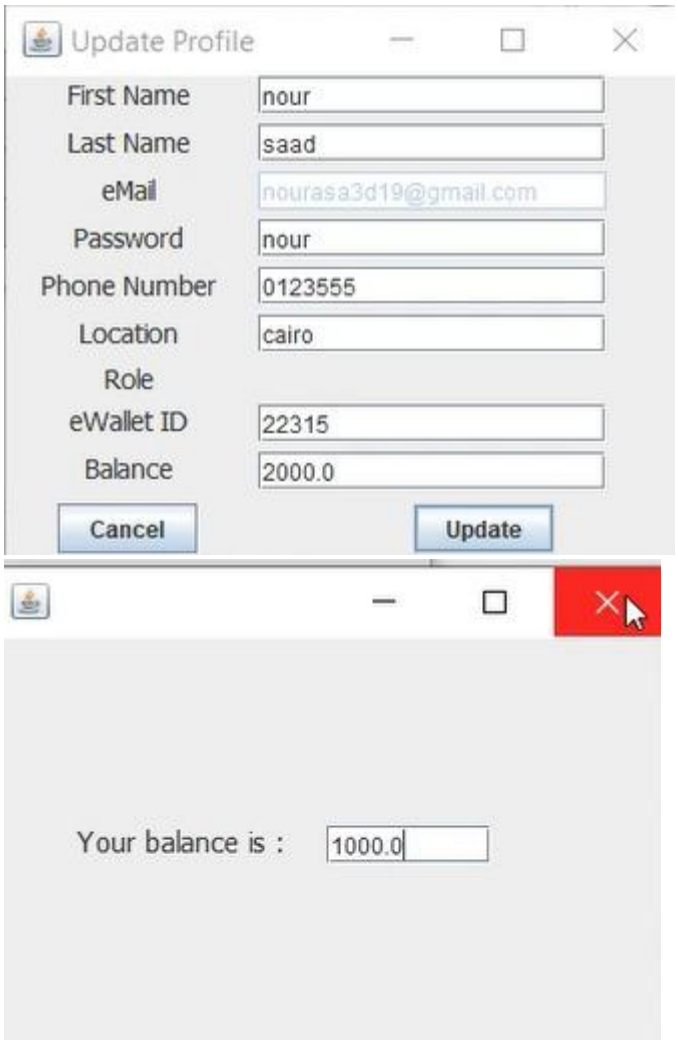


CS251: Phase 2 – <Team Name>
Project: <Project Name>

Software Design Specification



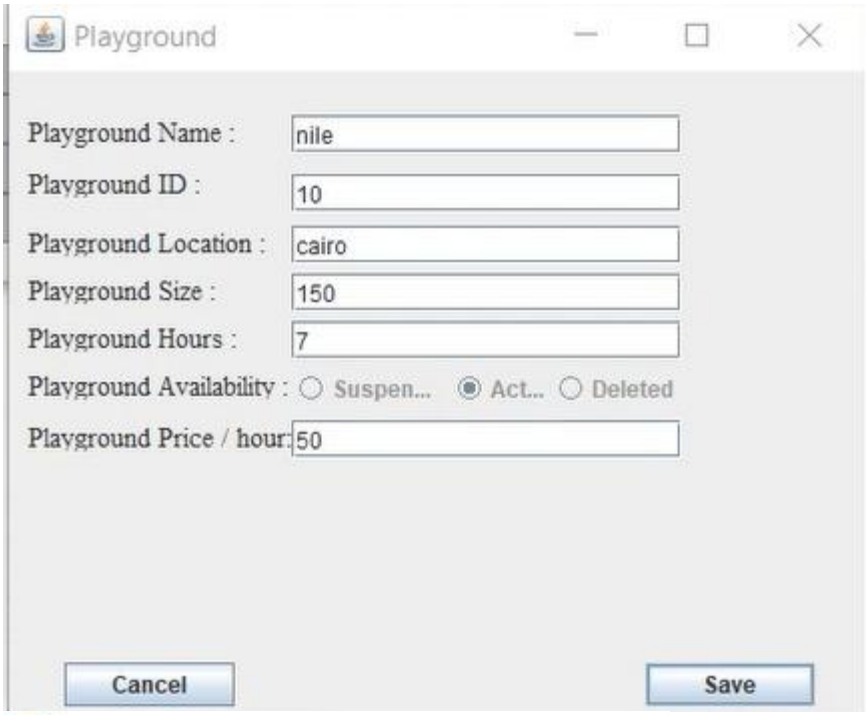
Software Design Specification

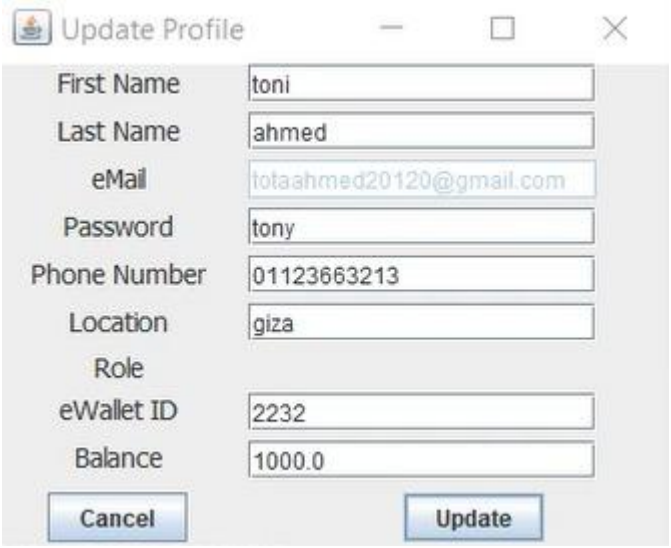


Playgroundowner options

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

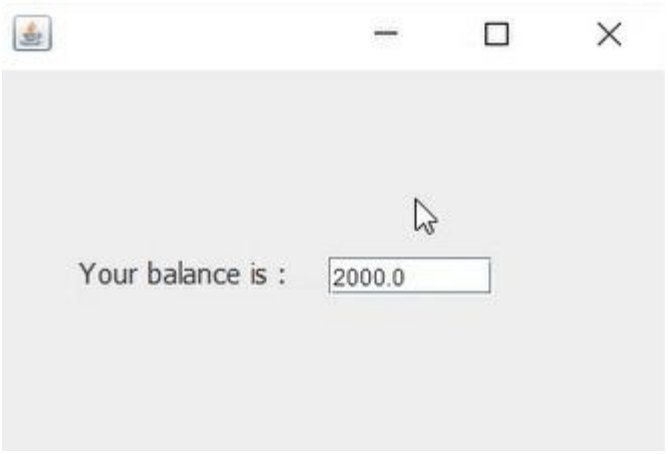
Software Design Specification





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

Software Design Specification



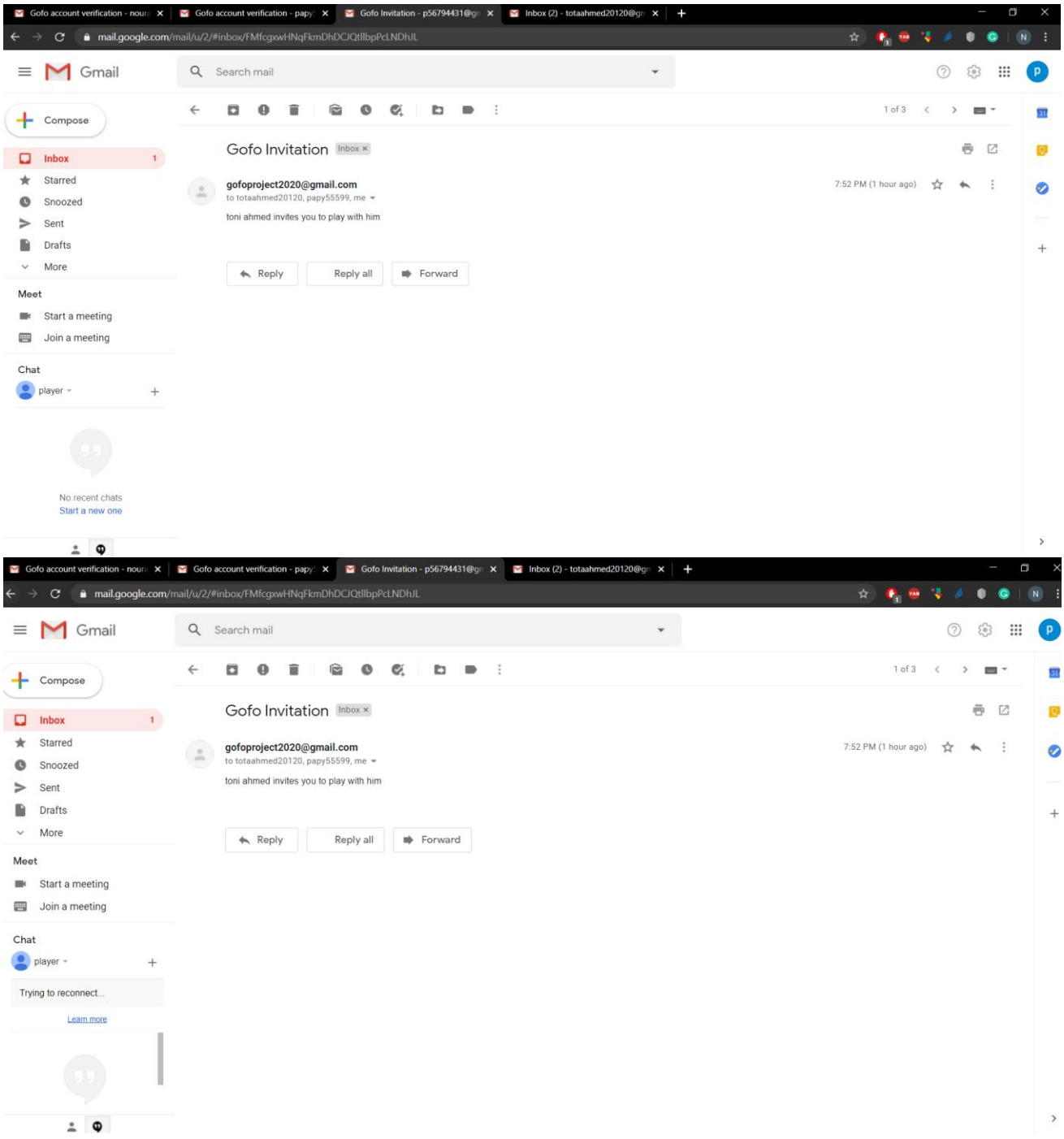
CS251: Phase 2 – <Team Name>
Project: <Project Name>

Software Design Specification

Verficiationcode for invite team

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

Software Design Specification



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

Software Design Specification

Link drive:
https://drive.google.com/file/d/1JCmw9en4fOOW95GuxnbT2dJpzik32fCk/view?usp=sharing
Link github:
https://github.com/Neimat1/Gofo.git
link video:
https://drive.google.com/file/d/1qholTFbEJgtD88iXmzmxglEqttN1utwf/view?usp=sharing

Authors

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