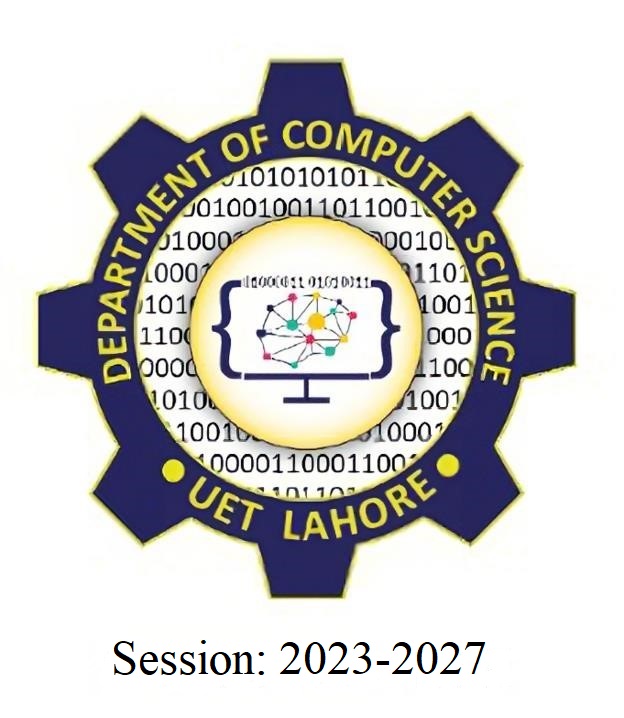
**Pysantas Medi Harmony**



**Submitted by:**

Muhammad Owais 2023-CS-131

**Supervised by:**

Sir Laeeq Khan Niazi

Department of Computer Science

**University of Engineering and Technology Lahore Pakistan**

Table of Contents

[ Contribution toward CS: 4](#_Toc153630928)

[ Why Medi Harmony: 4](#_Toc153630929)

[ What to Expect from Medi Harmony: 5](#_Toc153630930)

[❖ User Types on Medi Harmony: 5](#_Toc153630931)

[I. Admin: 5](#_Toc153630932)

[II. Pharmacist: 5](#_Toc153630933)

[III. Patient: 5](#_Toc153630934)

[IV. Doctor: 5](#_Toc153630935)

[V. Equipment manager: 5](#_Toc153630936)

[❖ Functional Requirements of Medi Harmony: 5](#_Toc153630937)

[❖ Wireframe of Medi Harmony: 8](#_Toc153630938)

[ Basic (For All types of Users): 8](#_Toc153630939)

[ Startup Interface: 8](#_Toc153630940)

[ Branch Menu: 8](#_Toc153630941)

[ Islamabad Branch Interface: 10](#_Toc153630942)

[ Lahore Branch Interface 10](#_Toc153630943)

[ Multan Branch Interface: 11](#_Toc153630944)

[ Login Menu: 11](#_Toc153630945)

[ Signup Menu 12](#_Toc153630946)

[ Sign in menu: 12](#_Toc153630947)

[ Admin Menu: 13](#_Toc153630948)

[ Add Patient: 13](#_Toc153630949)

[ View Patient: 14](#_Toc153630950)

[ Discharge Patient: 14](#_Toc153630951)

[ Profit Loss Calculation 15](#_Toc153630952)

[ Changing Salary: 15](#_Toc153630953)

[ Add Bed: 16](#_Toc153630954)

[ Add Doctor: 16](#_Toc153630955)

[ View Doctor: 17](#_Toc153630956)

[ Add Equipment-Manager: 17](#_Toc153630957)

[ Change Equipment-Manager: 18](#_Toc153630958)

[ Update Patient: 18](#_Toc153630959)

[ Doctor Menu: 19](#_Toc153630960)

[ View Admitted Patient: 19](#_Toc153630961)

[ View Available Beds: 20](#_Toc153630962)

[ View Doctor’s List: 20](#_Toc153630963)

[ Patient Menu: 21](#_Toc153630964)

[ Select Doctor: 21](#_Toc153630965)

[ Pay Bills: 22](#_Toc153630966)

[ View Prescription: 22](#_Toc153630967)

[ Patient Review: 23](#_Toc153630968)

[ Buy Medicines: 23](#_Toc153630969)

[ Pharmacist Menu: 24](#_Toc153630970)

[ Add Medicine: 24](#_Toc153630971)

[ View Medicines: 25](#_Toc153630972)

[ Update Medicine: 25](#_Toc153630973)

[ Expired medicine: 26](#_Toc153630974)

[ Mostly Used Medicines: 26](#_Toc153630975)

[ Remove Medicines: 27](#_Toc153630976)

[ Add Employ: 27](#_Toc153630977)

[ Remove Employ: 28](#_Toc153630978)

[ View Employ: 28](#_Toc153630979)

[ Equipment Manager: 29](#_Toc153630980)

[ Add Equipment: 29](#_Toc153630981)

[ View Equipment: 30](#_Toc153630982)

[ Remove Equipment: 30](#_Toc153630983)

[❖ Data Structure: 31](#_Toc153630984)

[❖ Function Prototypes: 38](#_Toc153630985)

[❖ Code: 40](#_Toc153630986)

[❖ Weakness: 127](#_Toc153630987)

[❖ Future Directions: 127](#_Toc153630988)

Table of Figures

[Figure 1. Startup Interface 10](#_Toc153863875)

[Figure 2. Islamabad Branch Interface 10](#_Toc153863876)

[Figure 3-Lahore Branch Interface 11](#_Toc153863877)

[Figure 4-Multan Branch Interface 11](#_Toc153863878)

[Figure 5- Login Menu- 12](#_Toc153863879)

[Figure 6-Sign up menu 13](#_Toc153863880)

[Figure 7-Signin Menu 13](#_Toc153863881)

[Figure 8-Admin Menu 14](#_Toc153863882)

[Figure 9-Add Patient Menu 14](#_Toc153863883)

[Figure 10-View Patient Menu 15](#_Toc153863884)

[Figure 11-Discharge Menu 15](#_Toc153863885)

[Figure 12-Profit Loss Menu 16](#_Toc153863886)

[Figure 13-Changing Salary Menu 16](#_Toc153863887)

[Figure 14-Add Bed Menu 17](#_Toc153863888)

[Figure 15-Add Doctor Menu 17](#_Toc153863889)

[Figure 16-View Doctor Menu 18](#_Toc153863890)

[Figure 17-Add Manager Menu 18](#_Toc153863891)

[‘ Figure 18-Change Manger Menu 19](#_Toc153863892)

[Figure 19-Update Patient Menu 19](#_Toc153863893)

[Figure 20-Doctor Menu 20](#_Toc153863894)

[Figure 21-Add Doctor Menu 20](#_Toc153863895)

[Figure 22-View Available Beds Menu 21](#_Toc153863896)

[Figure 23-View Available Doctor Menu 21](#_Toc153863897)

[Figure 24-Patient Menu 22](#_Toc153863898)

[Figure 25-Select Doctor Menu 22](#_Toc153863899)

[Figure 26-Bill Menu 23](#_Toc153863900)

[Figure 27-View Prescription Menu 23](#_Toc153863901)

[Figure 28-Patient Review Menu 24](#_Toc153863902)

[Figure 29-Buy Medicines Menu 24](#_Toc153863903)

[Figure 30-Pharmacist Menu 25](#_Toc153863904)

[Figure 31-Add Medicine Menu 25](#_Toc153863905)

[Figure 32-View Medicine Menu 26](#_Toc153863906)

[Figure 33-Update Medicine Menu 26](#_Toc153863907)

[Figure 34-Expired Medicine Menu 27](#_Toc153863908)

[Figure 35-Medicine in used Menu 27](#_Toc153863909)

[Figure 36-Remove Medicine Menu 28](#_Toc153863910)

[Figure 37-Add Employ Menu 28](#_Toc153863911)

[Figure 38-Remove Employ Menu 29](#_Toc153863912)

[Figure 39-View Employ Menu 29](#_Toc153863913)

[Figure 40-Equipment Manager Menu 30](#_Toc153863914)

[Figure 41-Add Equipment Menu 30](#_Toc153863915)

[Figure 42-View Equipment Menu 31](#_Toc153863916)

[Figure 43-Remove Equipment Menu 31](#_Toc153863917)

❖ **About Medi Harmony:**

Medi Harmony is a user-friendly application, built to help the patient, doctors, equipment manager, admin and pharmacist to approach different contents in the hospital. As the users can see the progress of the all the things like stock etc. They can also view the details of the equipment’s. The patient can take appointment, view medicines and all the other things. The patient sees the available beds in the hospital, see the bills online and pays it. Give the satisfactory report, buy medicines. The pharmacist adds, delete and view medicines stock. Also, he sees the expired medicines details from his stock.

##  Contribution toward CS:

Medi Harmony is a business application that can helps the patient to take appointment and to save his/her own precocious time. This application run the whole application in C++ and it will run accurately. This is the main thing that I can run all the things in this as a computer scientist. This application is capable of running a whole hospital management system easily.

##  Why Medi Harmony:

Medi Harmony a one of the credible applications all over country people like to work and share things with it. As the application is running very smoothly and there is no chance of fraud and any other misunderstanding as the is a big privacy before all the work. There is a copy right mark in this Medi Harmony.

##  What to Expect from Medi Harmony:

The Medi Harmony is a big application and trustable so feel free to contact us. Also, you can register his/her own bed in this application. So, the chance of misunderstanding will be ended. This is why, the major population of our country using this application.

# ❖ User Types on Medi Harmony:

Medi Harmony provides different access for different type of users. The users are divided into three categories, patient, pharmacist and Owner (Admin). Each user type has access to different types of command related to their need and requirements. User can login and verify their type by inputting the username and password for authentication.

The hierarchy and functionality of user types is as under: -

## I. Admin:

Owner or admin has access to add patients, to view the patients and add the patient, discharging the patient, add beds, add doctors, viewing the doctors list, add and remove the manager, change the salary and view the profit and loss. Also, he has to access to all the internal functions of the hospital.

## II. Pharmacist:

Pharmacist has access to add medicines, remove medicines, view medicines list, updating the medicines list, view the expired medicines, add the employ to the shop and remove the employ. He also gave prescription to some common diseases. He also views the medicines which are in demand all over the country.

## III. Patient:

Patient is the main user of the application. The main purpose of this application is to serve the patient as much as a possible. Patient have ability to view doctors schedule, available beds, selecting doctors, bill check, buying medicines. Also, patient can view the doctor schedule. Patient also gives the feedback to the staff if he likes the medication.

## IV. Doctor:

Doctor can view the patients added in the hospital and he also view that he hired by someone or not. To suggest the patient medication, he can also view the number of beds available in the hospital.

## V. Equipment manager:

There’s a manager in the hospital that can has the details of the equipment. He adds the equipment in the stock, view the admitted equipment, remove the broken equipment from the system. He also updated the added equipment.

# ❖ Functional Requirements of Medi Harmony:

Some of the functional requirements expected from Medi Harmony are as under:

|  |  |  |
| --- | --- | --- |
| ***User Type*** | ***Required Function to be Performed*** | ***Result of Action Performed*** |
| ***ADMIN*** | Add Patient | Admin can add the patient which come for treatment. |
| View Patient | Admin can also view the patients list which were added. |
| Discharge Patient | Admin has ability to discharge the patient after treatment. |
| Profit | Admin can view the profit by simply adding the patient. |
| Changing salary | Admin can change the salary of the doctors as well as pharmacist and manager. |
| Add Bed | Admin can add new beds if needed. For the suffering patients. |
| Add Doctors | Admin can add the doctors for the patients. |
| View Doctors List | Admin has ability to view the doctor’s list. |
| Add Equipment’s Manager | Admin can add manager. |
| Update patient | Admin have ability to update patient if any wrong information is entered. |
| ***Pharmacist*** | Add Medicine Form | Pharmacist can add the medicine. |
| View Medicines | Pharmacist can view the medicines list. |
| Update Medicines Form | Pharmacist can also view the medicines list. |
| View Employ | Pharmacist has ability to view the employ in the pharmacy department. |

|  |  |  |
| --- | --- | --- |
|  | Medicines in Demand | Pharmacist can view the medicines mostly used for diseases. |
|  | Expired Medicines | Pharmacist can also view the expired medicines. |
| Remove Medicines | Pharmacist can delete and exchange the unused medicines. |
| Add Employ | Pharmacist can also add the employ. |
| Remove Employ | Pharmacist can also remove employ. |
| ***Patient*** | View Doctors Schedule | Patient can view the doctors of the hospital. |
| View Prescription | Patient can also view the prescription to some diseases. |
| Billing and invoice | Patient can pay bills for the treatment. |
| View doctors list | Patient can view the doctors list. |
| Available Beds | Patient can see the beds. |
| Buy Medicines | Patient can buy the medicines. |
| ***Doctor*** | View Patient | Doctor can view patient which come for cure |
| View Available beds | Doctor can view the available beds to suggest patient. |
| View Doctor’s List | Doctor can view record list to know whether he is hired by someone or not. |
| ***Equipment Manager*** | Add Equipment | Manager can add the equipment for the surgery of the patients. |
| View Equipment | Manager can also access to the equipment record. |
| Remove Equipment | Manager can remove the equipment from the stock. |
| Update Medicines | Manager can also exchange the equipment which he wants. |

# ❖ Wireframe of Medi Harmony:

The following is the wireframe of Medi Harmony.

##  Basic (For All types of Users):

### Startup Interface:

Login Menu:



Figure 5- Login Menu-

* Signup Menu**:**



Figure 6-Sign up menu

### Sign in menu:



Figure 7-Signin Menu

### Admin Menu:



Figure 8-Admin Menu

### Add Patient:

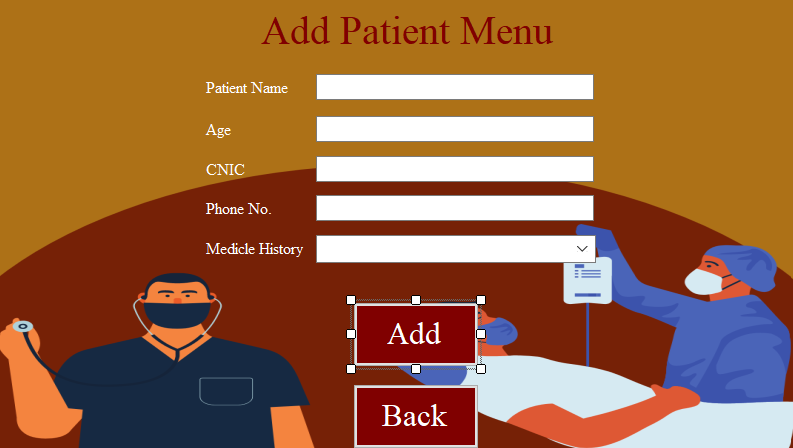
****

Figure 9-Add Patient Menu

### View Patient:



Figure 10-View Patient Menu

### Discharge Patient:

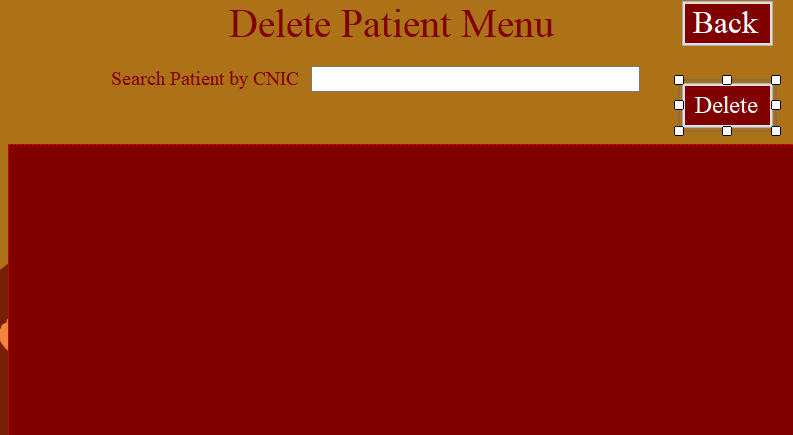


Figure 11-Discharge Menu

* Profit Loss Calculation**:**



Figure 12-Profit Loss Menu

**Add Bed:**

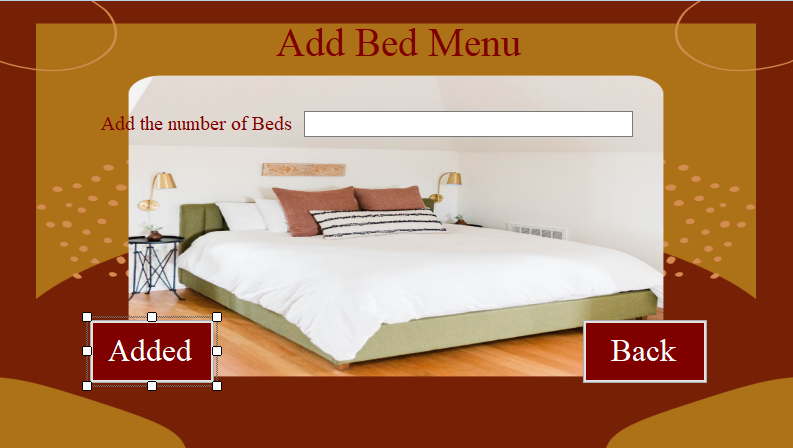


Figure 14-Add Bed Menu

### Add Doctor:



Figure 15-Add Doctor Menu

### View Doctor:

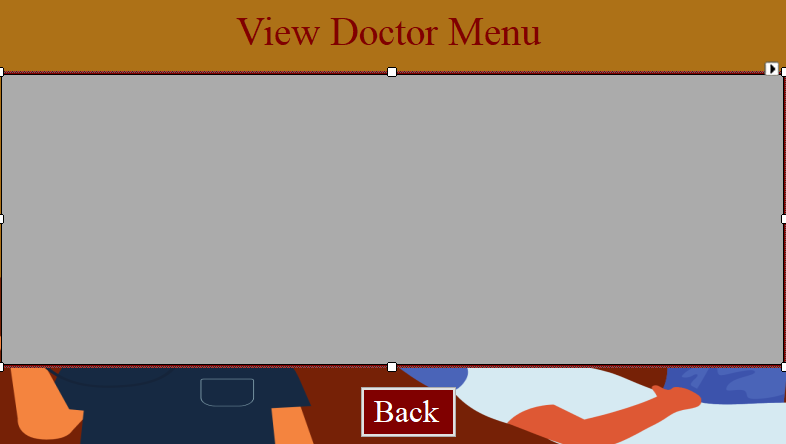


Figure 16-View Doctor Menu

### View Admitted Patient:

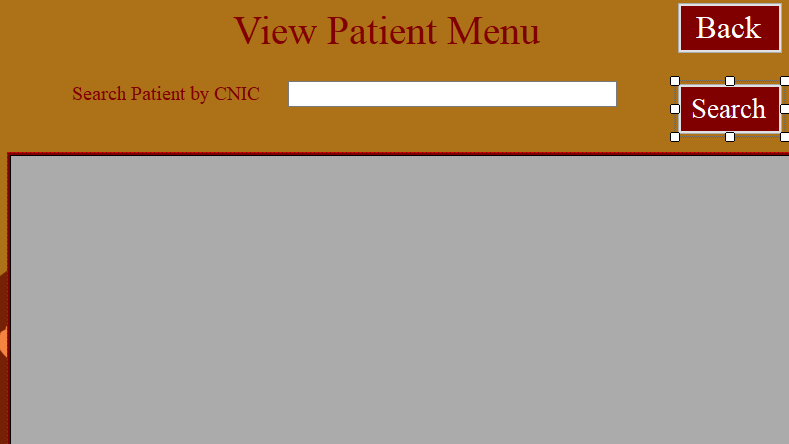


Figure 21-Add Doctor Menu

### View Doctor’s List:

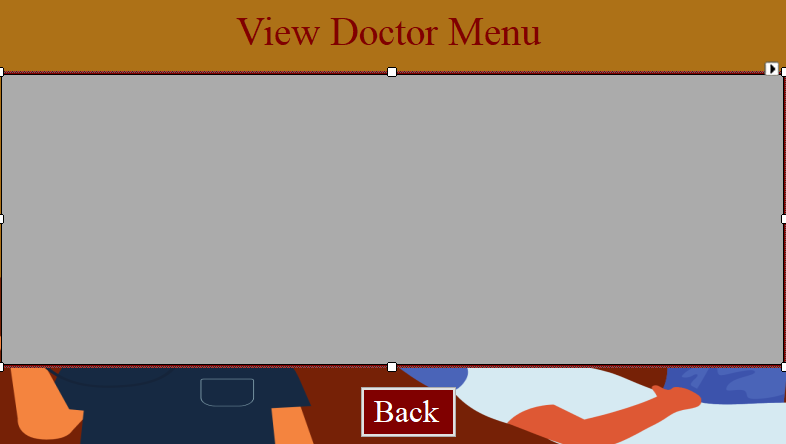


Figure 23-View Available Doctor Menu

### Patient Menu:



Figure 24-Patient Menu

### Select Doctor:



Figure 25-Select Doctor Menu

### Pay Bills:



Figure 26-Bill Menu

### View Prescription:

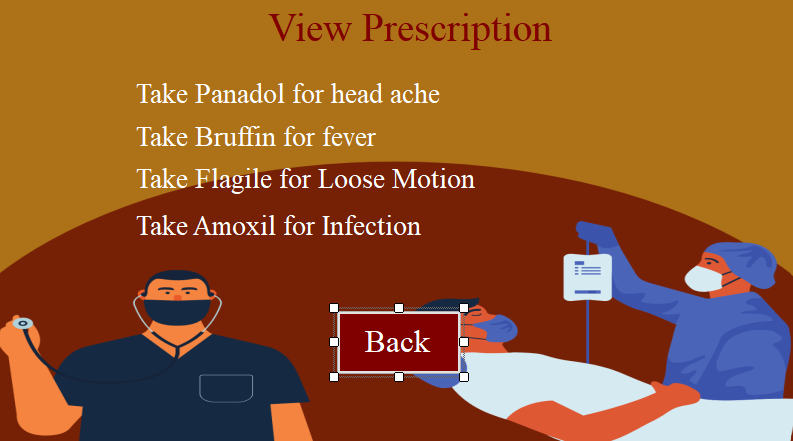


Figure 27-View Prescription Menu

### Patient Review:



Figure 28-Patient Review Menu

### Pharmacist Menu:



Figure 30-Pharmacist Menu

### Add Medicine:



Figure 31-Add Medicine Menu

### View Medicines:



Figure 32-View Medicine Menu

### Update Medicine:



Figure 33-Update Medicine Menu

### Mostly Used Medicines:



Figure 35-Medicine in used Menu

### Remove Medicines:

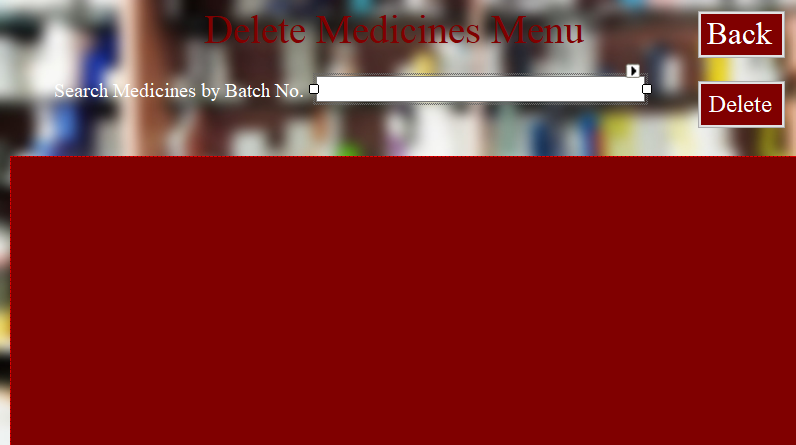


Figure 36-Remove Medicine Menu

### Add Employ:



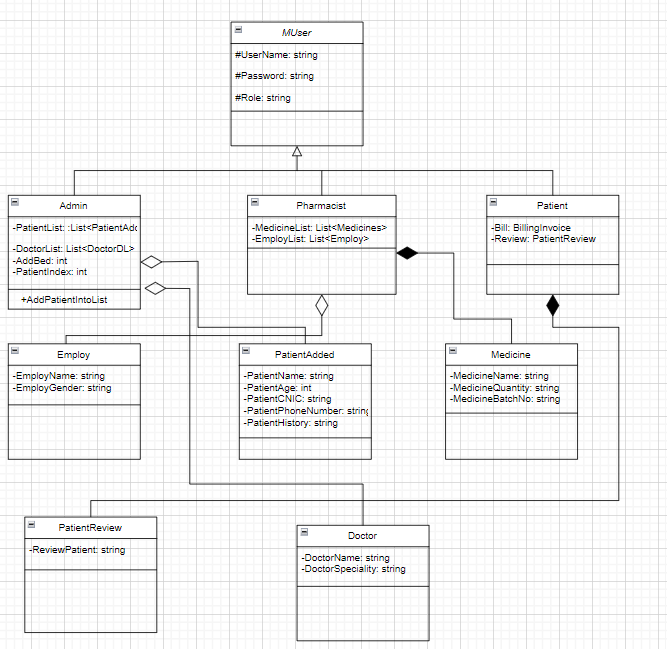
Figure 37-Add Employ Menu

### View Employ:



Figure 39-View Employ Menu

# ❖ CRC Card:



# ❖ Code:

# 

# ❖ BL:

using MyLibrary.BL;

using MyLibrary.DL\_Interface;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MyLibrary.Utilitys;

using MyLibrary.DL.FH;

namespace MyLibrary.DL.DB

{

public class MUserDL : IUser

{

private string conn;

public MUserDL(string conn)

{

this.conn=conn;

}

public bool AddData(MUserBL userBL)

{

string query = String.Format("Insert into MUser(Name, Gender, UserName, Password, Role)" + " values('{0}','{1}','{2}','{3}','{4}')", userBL.GetName(), userBL.GetGender(), userBL.GetUserName(), userBL.GetPassword(), userBL.GetRole());

SqlConnection sqlConnection = new SqlConnection(conn);

sqlConnection.Open();

SqlCommand sql = new SqlCommand(query, sqlConnection);

int rowsaffected = sql.ExecuteNonQuery();

sqlConnection.Close();

if (rowsaffected > 0)

{

return true;

}

else

{

return false;

}

}

public MUserBL SignIn(MUserBL user)

{

List<MUserBL> Users = RetrieveData();

foreach (MUserBL storedusers in Users)

{

if (storedusers.GetUserName() == user.GetUserName() && storedusers.GetPassword() == user.GetPassword())

{

return storedusers;

}

}

return null;

}

public List<MUserBL> RetrieveData()

{

List<MUserBL> users = new List<MUserBL>();

string query = "Select \* from MUser ";

SqlConnection sqlConnection = new SqlConnection(conn);

sqlConnection.Open();

SqlCommand sqlCommand = new SqlCommand(query, sqlConnection);

SqlDataReader reader = sqlCommand.ExecuteReader();

while (reader.Read())

{

MUserBL user = new MUserBL(Convert.ToString(reader["Name"]), Convert.ToString(reader["Gender"]), Convert.ToString(reader["Username"]), Convert.ToString(reader["Password"]), Convert.ToString(reader["Role"]));

users.Add(user);

}

reader.Close();

sqlConnection.Close();

return users;

}

public bool ValidateUser(MUserBL user)

{

List<MUserBL> Users = RetrieveData();

for (int i = 0; i < Users.Count; i++)

{

if (user.GetPassword() == Users[i].GetPassword() && user.GetUserName() == Users[i].GetUserName())

{ return true; }

}

return false;

}

}

}

# 

# ❖ DL with File Handling:

using MyLibrary.BL;

using MyLibrary.DL\_Interface;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MyLibrary.DL.FH

{

public class MUserFH

{

public class UserFH : IUser

{

private static List<MUserBL> users = new List<MUserBL>();

string path;

public UserFH(string path)

{

this.path = path;

users = RetrieveData();

}

public bool AddData(MUserBL user)

{

bool val = ValidateUser(user);

users.Add(user);

StreamWriter streamWriter = new StreamWriter(path, true);

streamWriter.WriteLine(user.GetName() + "," + user.GetGender() + "," + user.GetUserName() + "," + user.GetPassword() + "," + user.GetRole());

streamWriter.Flush();

streamWriter.Close();

return true;

}

public List<MUserBL> GetUsers()

{

return users;

}

public List<MUserBL> RetrieveData()

{

StreamReader streamReader = new StreamReader(path);

if (File.Exists(path))

{

string record;

while ((record = streamReader.ReadLine()) != null)

{

string name = GetField(record, 1);

string gender = GetField(record, 2);

string username = GetField(record, 3);

string password = GetField(record, 4);

string role = GetField(record, 5);

MUserBL user = new MUserBL(name, gender, username, password, role);

users.Add(user);

}

streamReader.Close();

}

return users;

}

public MUserBL SignIn(MUserBL user)

{

foreach (MUserBL storedusers in users)

{

if (storedusers.GetUserName() == user.GetUserName() && storedusers.GetPassword() == user.GetPassword())

{

return storedusers;

}

}

return null;

}

public bool ValidateUser(MUserBL user)

{

for (int i = 0; i < users.Count; i++)

{

if (users[i].GetUserName() == user.GetUserName() && users[i].GetPassword() == user.GetPassword())

{

return true;

}

}

return false;

}

private string GetField(string record, int field)

{

int commaCount = 1;

string result = "";

for (int x = 0; x < record.Count(); x++)

{

if (record[x] == ',')

{

commaCount++;

}

else if (commaCount == field)

{

result += record[x];

}

}

return result;

}

}

}

}

# ❖ DL with Data Base:

using MyLibrary.BL;

using MyLibrary.DL\_Interface;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MyLibrary.Utilitys;

using MyLibrary.DL.FH;

namespace MyLibrary.DL.DB

{

public class MUserDL : IUser

{

private string conn;

public MUserDL(string conn)

{

this.conn=conn;

}

public bool AddData(MUserBL userBL)

{

string query = String.Format("Insert into MUser(Name, Gender, UserName, Password, Role)" + " values('{0}','{1}','{2}','{3}','{4}')", userBL.GetName(), userBL.GetGender(), userBL.GetUserName(), userBL.GetPassword(), userBL.GetRole());

SqlConnection sqlConnection = new SqlConnection(conn);

sqlConnection.Open();

SqlCommand sql = new SqlCommand(query, sqlConnection);

int rowsaffected = sql.ExecuteNonQuery();

sqlConnection.Close();

if (rowsaffected > 0)

{

return true;

}

else

{

return false;

}

}

public MUserBL SignIn(MUserBL user)

{

List<MUserBL> Users = RetrieveData();

foreach (MUserBL storedusers in Users)

{

if (storedusers.GetUserName() == user.GetUserName() && storedusers.GetPassword() == user.GetPassword())

{

return storedusers;

}

}

return null;

}

public List<MUserBL> RetrieveData()

{

List<MUserBL> users = new List<MUserBL>();

string query = "Select \* from MUser ";

SqlConnection sqlConnection = new SqlConnection(conn);

sqlConnection.Open();

SqlCommand sqlCommand = new SqlCommand(query, sqlConnection);

SqlDataReader reader = sqlCommand.ExecuteReader();

while (reader.Read())

{

MUserBL user = new MUserBL(Convert.ToString(reader["Name"]), Convert.ToString(reader["Gender"]), Convert.ToString(reader["Username"]), Convert.ToString(reader["Password"]), Convert.ToString(reader["Role"]));

users.Add(user);

}

reader.Close();

sqlConnection.Close();

return users;

}

public bool ValidateUser(MUserBL user)

{

List<MUserBL> Users = RetrieveData();

for (int i = 0; i < Users.Count; i++)

{

if (user.GetPassword() == Users[i].GetPassword() && user.GetUserName() == Users[i].GetUserName())

{ return true; }

}

return false;

}

}

}