**NADRA Management System**



Session: 2023 – 2027

**Submitted by:**

Usman Ali Ashraf 2023-CS-106

**Supervised by:**

Mam Maida Mirza

Sir Laeeq uz Zaman Khan Niazi

Department of Computer Science

**University of Engineering and Technology, Lahore**

Table of Contents

Overview1

User Roles1

Admin1

Applicant1

**Features2**

**Technical Details2**

**Functional Requirement2**

**Non-Functional Requirements2**

**CRD3**

**Wireframes4**

**Code10**

****Overview:****

The NADRA (National Database and Registration Authority) Management System is a sophisticated and comprehensive software solution designed to efficiently manage and streamline the operations of NADRA offices across the country. Developed using cutting-edge technologies such as C# and the .NET Framework, this system caters to the diverse needs of NADRA administrators, officers, and citizens, providing a seamless and intuitive interface for various management tasks.

With a modular architecture, the NADRA Management System offers multiple interfaces, including Windows Forms, Console, and Library components. This design ensures adaptability to different user preferences and operational requirements, allowing for easy integration of new features and scalability to accommodate future expansions or modifications.

At its foundation, the NADRA Management System relies on a robust SQL database to securely store and manage vast amounts of critical data, including citizen records, biometric information, and administrative documents. Through efficient data handling mechanisms, the system ensures data integrity, confidentiality, and high-speed access, meeting the stringent security and performance requirements of NADRA's operations.

Furthermore, the system incorporates advanced file management capabilities to handle various documents and records essential to NADRA's functions, such as identity card applications, birth certificates, and citizenship records. With seamless integration of file handling features, administrators can effortlessly manage and retrieve documents, streamlining workflow processes and enhancing operational efficiency.

****User Roles:****

The user roles are given below:

Admin:

* Manage Member accounts (create, edit, delete, view)
* Manage NADRA Activities
* Manage All registered citizens
* Add another Admin
* Reply and View Complaints
* Manage Applications for Health Card

Applicant/User:

* Track Applications
* Apply for NADRA Activities
* View and print certificate
* Apply for Health Card
* Add Complaints and View Response
* Apply for CNIC and Covid Card

Features:

* User authentication and authorization
* User Management
* Citizen management (create, edit, delete, update)
* Progress tracking and reporting
* Allotment of CNIC and Covid Certificate
* Data backup and restore using file handling
* SQL database integration for data storage and retrieval
* Complaint Management
* User interface with console for easy debugging

Technical Details:

* Programming Language: C#
* Development Framework: .NET Framework
* Database: SQL Server
* File Handling: txt files
* User Interface: Windows Forms and Console

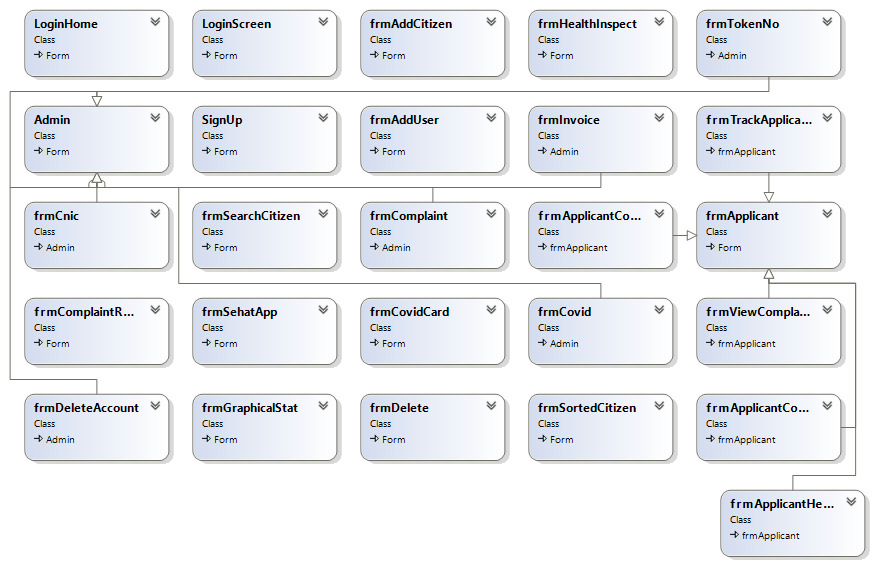
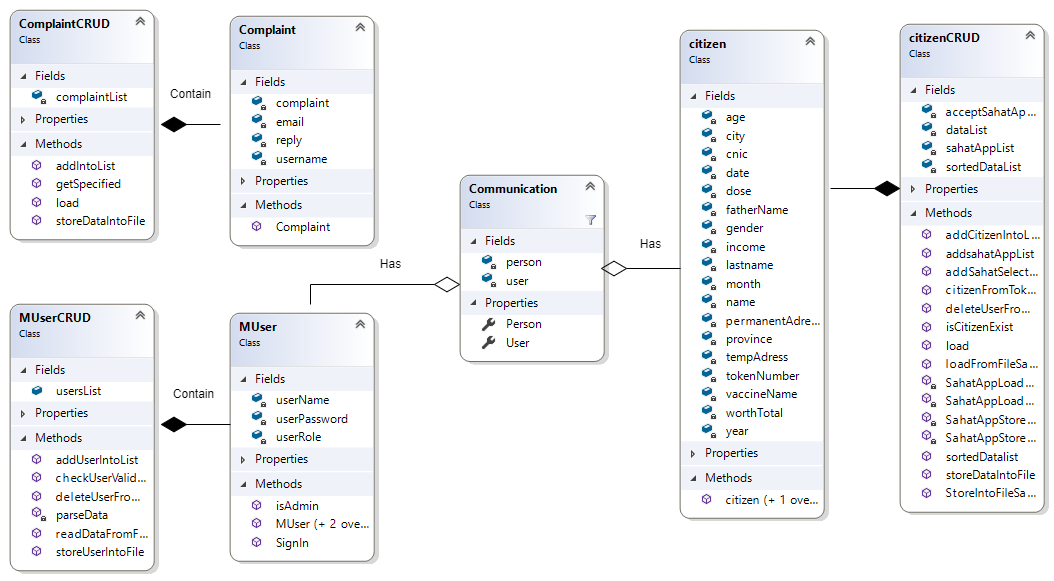
Functional Requirements:

* User authentication and authorization
* Citizen Management
* Application tracking and reporting
* System settings configuration
* Data backup and restore

Non-Functional Requirements:

* Security: Ensure user data privacy and security
* Performance: Optimize system performance for efficient data retrieval and storage
* Usability: Provide an intuitive and user-friendly interface for all users

**Class Diagrams**



Wireframes:

The wireframes of the app are given below:

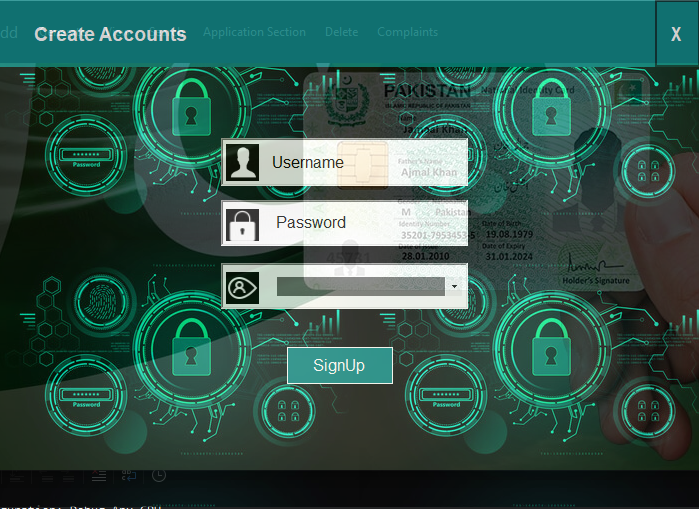
Login Page:

****

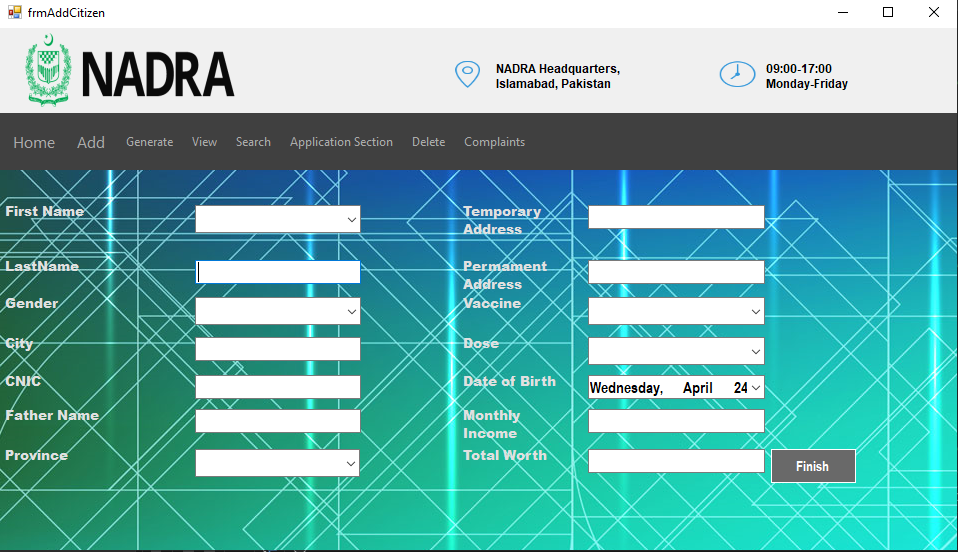
Admin HomePage:

****

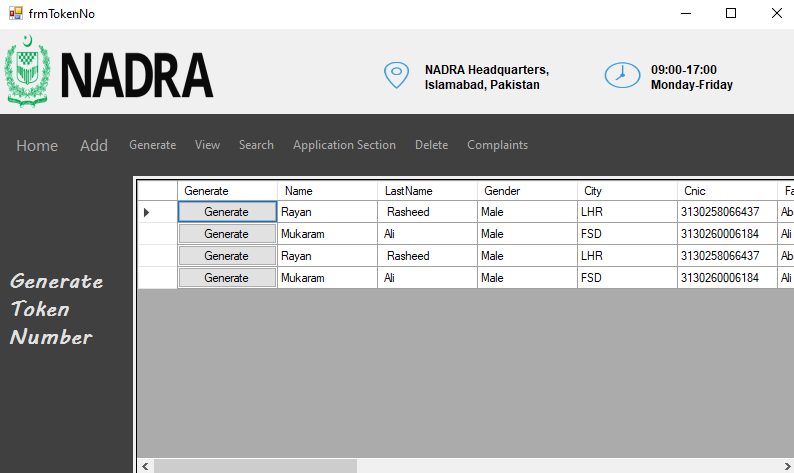
SignUp Page:

****

Add Citizen Page:

****

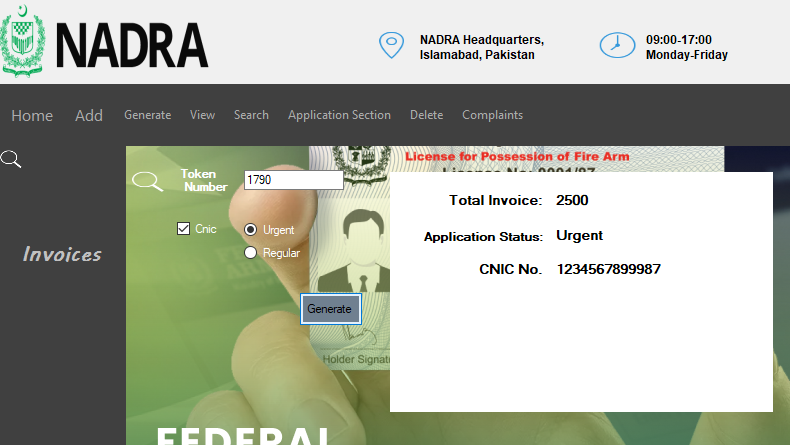
Generate Token No.:

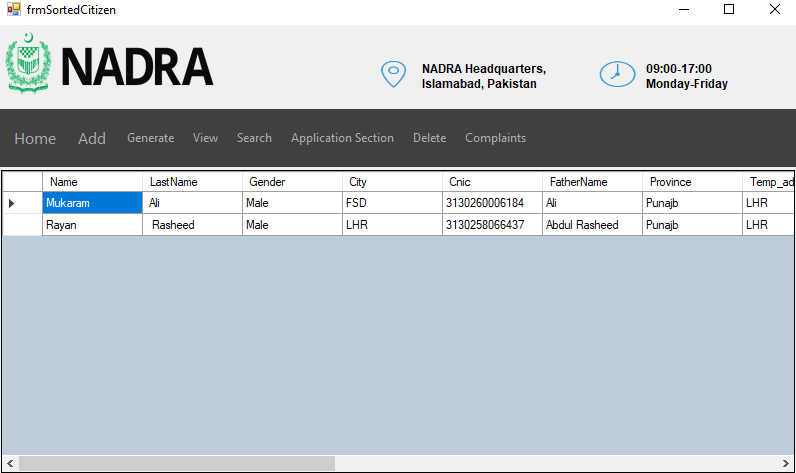


Generate CNIC:

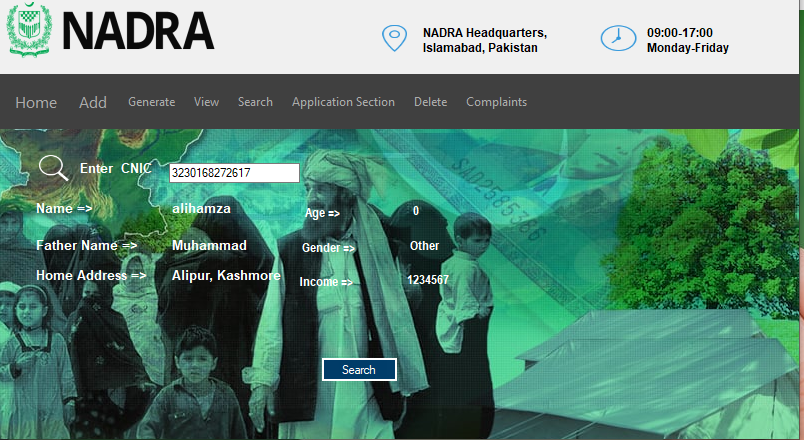


Generate Invoice:

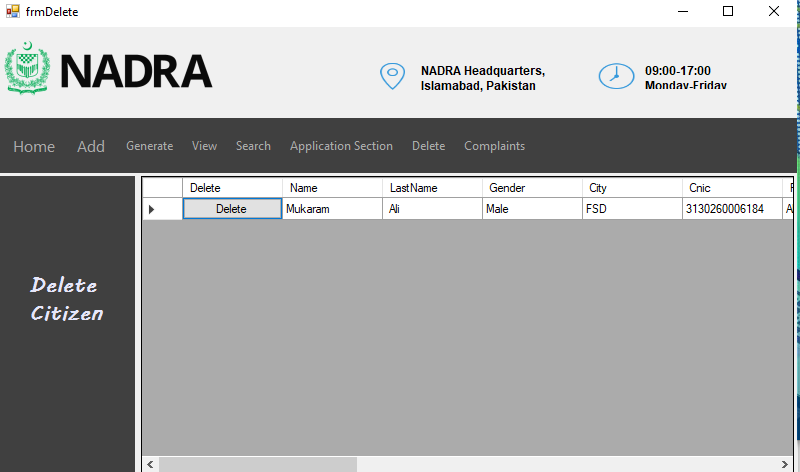
**View Sorted Citizens**:



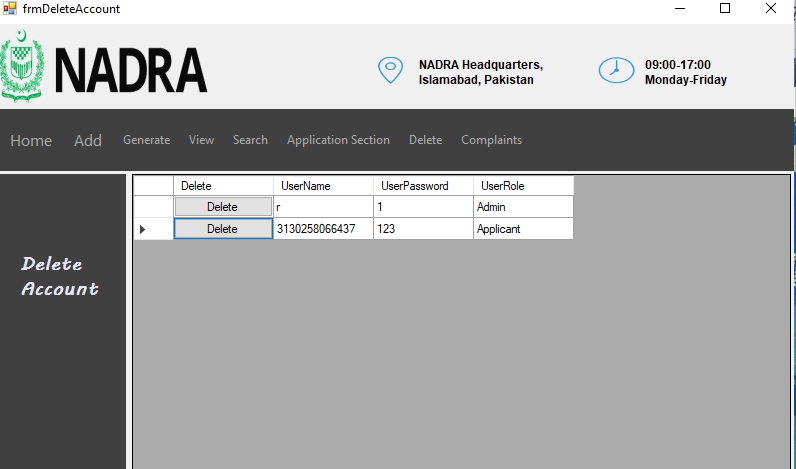
Search Citizens:

****

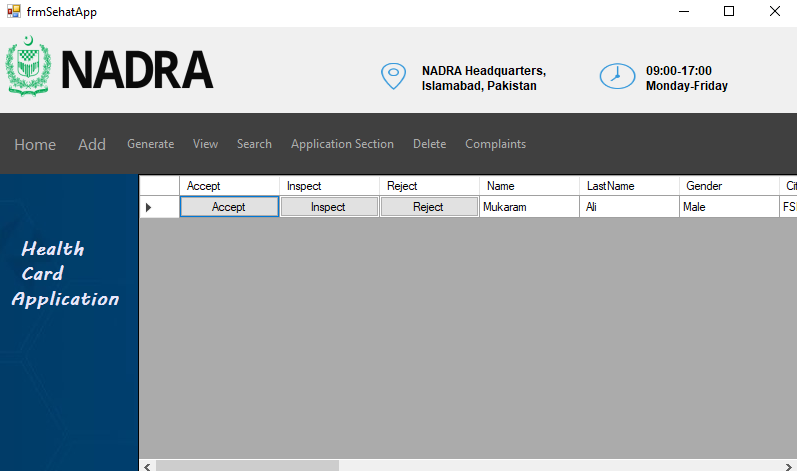
Delete Citizens:



Delete Users Accounts:



Manage Health Card Applications:



Inspect Health Card:



Track Application:



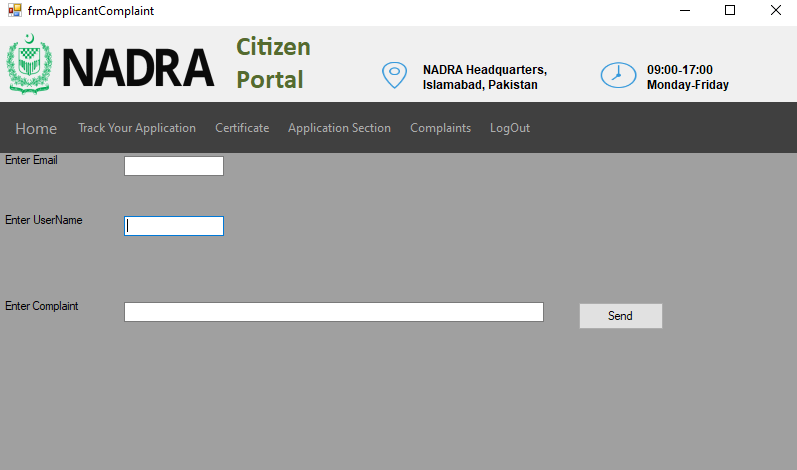
Covid Certificate:

****

Apply for Health Card:

****

Send Complaint:



Complaint Reply:



**CitizenBL**

public class citizen

{

private string name;

private string lastname;

private string gender;

private string city;

private string cnic;

private string fatherName;

private string province;

private string tempAdress;

private string permanentAdress;

private string vaccineName;

private int dose;

private int age;

private int year;

private int month;

private int date;

private int income;

private double worthTotal ;

private int tokenNumber;

public citizen(string name, string lastname, string gender, string city, string cnic, string fatherName, string province, string tempAdress, string permanentAdress, string vaccineName, int dose, int date, int month, int year, int income, double worthTotal)

{

this.name = name;

this.lastname = lastname;

this.gender = gender;

this.city = city;

this.cnic = cnic;

this.fatherName = fatherName;

this.province = province;

this.tempAdress = tempAdress;

this.permanentAdress = permanentAdress;

this.vaccineName = vaccineName;

this.dose = dose;

this.year = year;

this.month = month;

this.date = date;

this.income = income;

this.worthTotal = worthTotal;

}

public citizen(){

}

public string Name { get => name; set => name = value; }

public string LastName { get => lastname; set => lastname = value; }

public string Gender { get => gender; set => gender = value; }

public string City { get => city; set => city = value; }

public string Cnic { get => cnic; set => cnic = value; }

public string FatherName { get => fatherName; set => fatherName = value; }

public string Province { get => province; set => province = value; }

public string Temp\_adress { get => tempAdress; set => tempAdress = value; }

public string PermanentAdress { get => permanentAdress; set => permanentAdress = value; }

public string VaccineName { get => vaccineName; set => vaccineName = value; }

public int Dose { get => dose; set => dose = value; }

public int Age { get => age; set => age = value; }

public int TokenNumber { get => tokenNumber; set => tokenNumber = value; }

public int Year { get => year; set => year = value; }

public int Month { get => month; set => month = value; }

public int Date { get => date; set => date = value; }

public int Income { get => income; set => income = value; }

public double WorthTotal { get => worthTotal; set => worthTotal = value; }

}

**ComplaintBL**

public class Complaint

{

private string email;

private string username;

private string complaint;

private string reply;

public Complaint(string email,string username,string complaint)

{

this.email = email;

this.username = username;

this.complaint = complaint;

}

public string Name { get => username; set => username = value; }

public string Complaints { get => complaint; set => complaint = value; }

public string Email { get => email; set => email = value; }

public string Reply { get => reply; set => reply = value; }

}

**MUserBL**

public class MUser

{

private string userName;

private string userPassword;

private string userRole;

public string UserName { get => userName; set => userName = value; }

public string UserPassword { get => userPassword; set => userPassword = value; }

public string UserRole { get => userRole; set => userRole = value; }

public MUser(string userName, string userPassword, string userRole)

{

this.userName = userName;

this.userPassword = userPassword;

this.userRole = userRole;

}

public MUser()

{

}

public MUser(string userName, string userPassword)

{

this.userName = userName;

this.userPassword = userPassword;

this.userRole = "NA";

}

public bool isAdmin()

{

if (userRole == "Admin")

{

return true;

}

return false;

}

public static MUser SignIn(MUser user)

{

foreach (MUser storedUser in MUserCRUD.usersList)

{

if (storedUser.userName == user.userName && storedUser.userPassword == user.userPassword)

{

return storedUser;

}

}

return null;

}

}

class citizenCRUD

{

private static List<citizen> dataList = new List<citizen>();

private static List<citizen> sortedDataList = new List<citizen>();

private static List<citizen> sahatAppList = new List<citizen>();

private static List<citizen> acceptSahatAppList = new List<citizen>();

public static List<citizen> DataList { get => dataList; }

public static void addCitizenIntoList(citizen c)

{

dataList.Add(c);

}

public static void addSahatSelectedCitizen(citizen c)

{

acceptSahatAppList.Add(c);

}

public static void addsahatAppList(citizen c)

{

sahatAppList.Add(c);

}

static public void sortedDatalist()

{

sortedDataList = dataList.OrderByDescending(o => o.Age).ToList();

}

public static List<citizen> SortedDataList

{

get => sortedDataList;

}

public static List<citizen> AcceptSahatAppList

{

get => acceptSahatAppList;

}

public static List<citizen> SahatAppList

{

get => sahatAppList;set => sahatAppList = value;

}

public static void storeDataIntoFile(string path)

{

StreamWriter file = new StreamWriter(path, false);

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

{

if (i == dataList.Count - 1)

{

file.Write(dataList[i].Name + ", " + dataList[i].LastName + "," + dataList[i].Gender + "," + dataList[i].City + "," + dataList[i].Cnic + "," + dataList[i].FatherName + "," + dataList[i].Province + "," + dataList[i].Temp\_adress + "," + dataList[i].PermanentAdress + "," + dataList[i].VaccineName + "," + dataList[i].Dose + ","+ dataList[i].Date + ","+ dataList[i].Month + ","+ dataList[i].Year + "," + dataList[i].Income + "," + dataList[i].WorthTotal+","+dataList[i].Age+","+ dataList[i].TokenNumber);

}

else

{

file.WriteLine(dataList[i].Name + ", " + dataList[i].LastName + "," + dataList[i].Gender + "," + dataList[i].City + "," + dataList[i].Cnic + "," + dataList[i].FatherName + "," + dataList[i].Province + "," + dataList[i].Temp\_adress + "," + dataList[i].PermanentAdress + "," + dataList[i].VaccineName + "," + dataList[i].Dose + "," + dataList[i].Date + "," + dataList[i].Month + "," + dataList[i].Year + "," + dataList[i].Income + "," + dataList[i].WorthTotal + "," + dataList[i].Age + "," + dataList[i].TokenNumber);

}

}

file.Flush();

file.Close();

}

public static void load( string path)

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*laoding recorda from file\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

string line;

if (File.Exists(path))

{

StreamReader file = new StreamReader(path);

while (((line = file.ReadLine())) != null)

{

string[] record = line.Split(',');

citizen Add = new citizen(record[0], record[1], record[2], record[3], record[4], record[5], record[6], record[7], record[8], record[9], int.Parse(record[10]), int.Parse(record[11]), int.Parse(record[12]), int.Parse(record[13]), int.Parse(record[14]), double.Parse(record[15]));

Add.Age = int.Parse(record[16]);

Add.TokenNumber = int.Parse(record[17]);

citizenCRUD.addCitizenIntoList(Add);

// uploaading temporary arr1ay data into orignal array

}

file.Close();

}

}

public static citizen isCitizenExist(string cnic)

{

foreach(var person in dataList)

{

if (person.Cnic == cnic)

{

return person;

}

}

return null;

}

static public void deleteUserFromList(citizen user,List<citizen >person)

{

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

{

if (person[x].Cnic==user.Cnic)

{

person.RemoveAt(x);

}

}

}

public static void StoreIntoFileSahatApp(string path, string path1)

{

SahatAppStoreAppCitizen(path);

SahatAppStoreSelected(path1);

}

private static void SahatAppStoreAppCitizen(string path)

{

StreamWriter file = new StreamWriter(path, false);

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

{

if (i == sahatAppList.Count - 1)

{

file.Write(sahatAppList[i].Cnic);

}

else

{

file.WriteLine(sahatAppList[i].Cnic);

}

}

file.Flush();

file.Close();

}

private static void SahatAppStoreSelected(string path)

{

StreamWriter file = new StreamWriter(path, false);

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

{

if (i == acceptSahatAppList.Count - 1)

{

file.Write(acceptSahatAppList[i].Cnic);

}

else

{

file.WriteLine(acceptSahatAppList[i].Cnic);

}

}

file.Flush();

file.Close();

}

public static void loadFromFileSahatApp(string path,string path1)

{

SahatAppLoadAppCitizen(path);

SahatAppLoadSelected(path1);

}

private static void SahatAppLoadAppCitizen(string path)

class ComplaintCRUD

{

private static List<Complaint> complaintList = new List<Complaint>();

public static List<Complaint> ComplaintList

{

get => complaintList;

set => complaintList = value;

}

public static void addIntoList(Complaint c)

{

complaintList.Add(c);

}

public static void storeDataIntoFile(string path)

{

StreamWriter file = new StreamWriter(path, false);

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

{

if (i == ComplaintList.Count - 1)

{

file.Write(ComplaintList[i].Email + "," + ComplaintList[i].Name + ",", ComplaintList[i].Complaints);

}

else

{

file.WriteLine(ComplaintList[i].Email + "," + ComplaintList[i].Name + "," + ComplaintList[i].Complaints);

}

}

file.Flush();

file.Close();

}

public static void load(string path)

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*laoding recorda from file\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

string line;

if (File.Exists(path))

{

StreamReader file = new StreamReader(path);

while (((line = file.ReadLine())) != null)

{

string[] record = line.Split(',');

Complaint Add = new Complaint(record[0], record[1], record[2]);

addIntoList(Add);

// uploaading temporary arr1ay data into orignal array

}

file.Close();

}

}

public static Complaint getSpecified(MUser user)

{

foreach(var cmp in complaintList)

{

if (cmp.Name == user.UserName)

{

return cmp;

}

}

return null;

}

}

class MUserCRUD

{

public static List<MUser> usersList = new List<MUser>();

public static void addUserIntoList(MUser user)

{

usersList.Add(user);

}

public static List<MUser> UsersList { get => usersList; }

public static bool readDataFromFile(string path)

{

if (File.Exists(path))

{

StreamReader fileVariable = new StreamReader(path);

string record;

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

{

string userName = parseData(record, 1);

string userPassword = parseData(record, 2);

string userRole = parseData(record, 3);

MUser user = new MUser(userName, userPassword, userRole);

addUserIntoList(user);

}

fileVariable.Close();

return true;

}

return false;

}

static string parseData(string record, int field)

{

int comma = 1;

string item = "";

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

{

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

{

comma++;

}

else if (comma == field)

{

item = item + record[x];

}

}

return item;

}

static public void storeUserIntoFile( string path)

{

//StreamWriter file = new StreamWriter(path, true);

//file.WriteLine(user.UserName+ "," + user.UserPassword + "," + user.UserRole);

//file.Flush();

//file.Close();

StreamWriter file = new StreamWriter(path);

foreach (MUser storedUser in usersList)

{

file.WriteLine(storedUser.UserName + "," + storedUser.UserPassword + "," + storedUser.UserRole);

}

file.Flush();

file.Close();

}

static public MUser checkUserValidation(string userName,string password)

{

foreach(var user in usersList)

{

if (user.UserName == userName && user.UserPassword == password)

{

return user;

}

}

return null;

}

static public void deleteUserFromList(MUser user)

{

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

{

if (usersList[x].UserName == user.UserName && usersList[x].UserPassword == user.UserPassword && usersList[x].UserRole == user.UserRole)

{

usersList.RemoveAt(x);

}

}

}

class MUserCRUD

{

public static List<MUser> usersList = new List<MUser>();

public static void addUserIntoList(MUser user)

{

usersList.Add(user);

}

public static List<MUser> UsersList { get => usersList; }

public static bool readDataFromFile(string path)

{

if (File.Exists(path))

{

StreamReader fileVariable = new StreamReader(path);

string record;

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

{

string userName = parseData(record, 1);

string userPassword = parseData(record, 2);

string userRole = parseData(record, 3);

MUser user = new MUser(userName, userPassword, userRole);

addUserIntoList(user);

}

fileVariable.Close();

return true;

}

return false;

}

static string parseData(string record, int field)

{

int comma = 1;

string item = "";

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

{

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

{

comma++;

}

else if (comma == field)

{

item = item + record[x];

}

}

return item;

}

static public void storeUserIntoFile( string path)

{

//StreamWriter file = new StreamWriter(path, true);

//file.WriteLine(user.UserName+ "," + user.UserPassword + "," + user.UserRole);

//file.Flush();

//file.Close();

StreamWriter file = new StreamWriter(path);

foreach (MUser storedUser in usersList)

{

file.WriteLine(storedUser.UserName + "," + storedUser.UserPassword + "," + storedUser.UserRole);

}

file.Flush();

file.Close();

}

static public MUser checkUserValidation(string userName,string password)

{

foreach(var user in usersList)

{

if (user.UserName == userName && user.UserPassword == password)

{

return user;

}

}

return null;

}

static public void deleteUserFromList(MUser user)

{

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

{

if (usersList[x].UserName == user.UserName && usersList[x].UserPassword == user.UserPassword && usersList[x].UserRole == user.UserRole)

{

usersList.RemoveAt(x);

}

}

}

class MUserCRUD

{

public static List<MUser> usersList = new List<MUser>();

public static void addUserIntoList(MUser user)

{

usersList.Add(user);

}

public static List<MUser> UsersList { get => usersList; }

public static bool readDataFromFile(string path)

{

if (File.Exists(path))

{

StreamReader fileVariable = new StreamReader(path);

string record;

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

{

string userName = parseData(record, 1);

string userPassword = parseData(record, 2);

string userRole = parseData(record, 3);

MUser user = new MUser(userName, userPassword, userRole);

addUserIntoList(user);

}

fileVariable.Close();

return true;

}

return false;

}

static string parseData(string record, int field)

{

int comma = 1;

string item = "";

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

{

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

{

comma++;

}

else if (comma == field)

{

item = item + record[x];

}

}

return item;

}

static public void storeUserIntoFile( string path)

{

//StreamWriter file = new StreamWriter(path, true);

//file.WriteLine(user.UserName+ "," + user.UserPassword + "," + user.UserRole);

//file.Flush();

//file.Close();

StreamWriter file = new StreamWriter(path);

foreach (MUser storedUser in usersList)

{

file.WriteLine(storedUser.UserName + "," + storedUser.UserPassword + "," + storedUser.UserRole);

}

file.Flush();

file.Close();

}

static public MUser checkUserValidation(string userName,string password)

{

foreach(var user in usersList)

{

if (user.UserName == userName && user.UserPassword == password)

{

return user;

}

}

return null;

}

static public void deleteUserFromList(MUser user)

{

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

{

if (usersList[x].UserName == user.UserName && usersList[x].UserPassword == user.UserPassword && usersList[x].UserRole == user.UserRole)

{

usersList.RemoveAt(x);

}

}

}

public partial class frmAddCitizen : Form

{

public frmAddCitizen()

{

InitializeComponent();

}

private void cmdAdd\_Click(object sender, EventArgs e)

{

if (txtFName.Text!=""&& txtLastName.Text != ""&& cboGender.Text != ""&& txtCity.Text != ""&& txtCnic.Text != ""&& txtFatherName.Text != ""&& cboProvince.Text != ""&& txtTempAdress.Text != ""&& txtPermAdress.Text != ""&& cboVaccine.Text != ""&& cboDose.Text != ""&&dateTimePicker1.Text != ""&& txtIncome.Text != ""&& txtTotalWorth.Text != "")

{

int year = dateTimePicker1.Value.Year;

int month = dateTimePicker1.Value.Month;

int day = dateTimePicker1.Value.Day;

int Presentyear = DateTime.Now.Year;

citizen Add = new citizen(txtFName.Text, txtLastName.Text, cboGender.Text, txtCity.Text, txtCnic.Text, txtFatherName.Text, cboProvince.Text, txtTempAdress.Text, txtPermAdress.Text, cboVaccine.Text, int.Parse(cboDose.Text), day, month, year, int.Parse(txtIncome.Text), int.Parse(txtTotalWorth.Text));

Add.Age = Presentyear - year;

citizenCRUD.addCitizenIntoList(Add);

citizenCRUD.storeDataIntoFile(FilePath.dataPath);

Admin a = new Admin();

a.Show();

this.Hide();

}

else

{

MessageBox.Show("Make sure all field should be filled!");

}

}

private void userLoginToolStripMenuItem\_Click(object sender, EventArgs e)

{

frmAddUser form = new frmAddUser();

form.Show();

}

private void txtFName\_TextChanged(object sender, EventArgs e)

{

}

private void tableLayoutPanel1\_Paint(object sender, PaintEventArgs e)

{

}

private void searchToolStripMenuItem\_Click(object sender, EventArgs e)

{

public partial class frmAddUser : Form

{

public frmAddUser()

{

InitializeComponent();

}

private void cmdClose\_Click(object sender, EventArgs e)

{

this.Close();

}

private void cmdSignUp\_Click(object sender, EventArgs e)

{

MUser user = new MUser(txtUserName.Text, txtPassword.Text, cboUserRole.Text);

MUserCRUD.addUserIntoList(user);

MUserCRUD.storeUserIntoFile(FilePath.credentialPath);

MessageBox.Show("SuccessFully Done!");

this.Close();

}

}

public partial class frmAddUser : Form

{

public frmAddUser()

{

InitializeComponent();

}

private void cmdClose\_Click(object sender, EventArgs e)

{

this.Close();

}

private void cmdSignUp\_Click(object sender, EventArgs e)

{

MUser user = new MUser(txtUserName.Text, txtPassword.Text, cboUserRole.Text);

MUserCRUD.addUserIntoList(user);

MUserCRUD.storeUserIntoFile(FilePath.credentialPath);

MessageBox.Show("SuccessFully Done!");

this.Close();

}

}

public partial class frmApplicantComplaint : frmApplicant

{

public frmApplicantComplaint()

{

InitializeComponent();

}

private void tableLayoutPanel4\_Paint(object sender, PaintEventArgs e)

{

}

private void cmdSend\_Click(object sender, EventArgs e)

{

Complaint c = new Complaint(txtEmail.Text,txtUserName.Text, txtComplaint.Text);

ComplaintCRUD.addIntoList(c);

ComplaintCRUD.storeDataIntoFile(FilePath.complaintpath);

txtUserName.Text = "";

txtEmail.Text = "";

txtComplaint.Text="";

}

}

public partial class frmApplicantCovid : frmApplicant

{

public frmApplicantCovid()

{

InitializeComponent();

}

private void frmApplicantCovid\_Load(object sender, EventArgs e)

{

}

private void cmdSearch\_Click(object sender, EventArgs e)

{

citizen person = citizenCRUD.isCitizenExist(txtToken.Text);

if (person != null)

{

frmCovidCard form = new frmCovidCard(person);

form.Show();

}

else

{

MessageBox.Show("Not Registered!");

}

}

}

public partial class frmApplicantHealth : frmApplicant

{

public frmApplicantHealth()

{

InitializeComponent();

}

private void frmApplicantHealth\_Load(object sender, EventArgs e)

{

lblNameFront.Text = Communication.Person.Name;

}

private void button1\_Click(object sender, EventArgs e)

{

if (Communication.Person!=null){

citizenCRUD.SahatAppList.Add(Communication.Person);

}

}

}

public partial class frmComplaint : Admin

{

public frmComplaint()

{

InitializeComponent();

}

public void dataBind()

{

gvComplaint.DataSource = null;

gvComplaint.DataSource = ComplaintCRUD.ComplaintList;

gvComplaint.Refresh();

}

private void frmComplaint\_Load(object sender, EventArgs e)

{

gvComplaint.DataSource = ComplaintCRUD.ComplaintList;

}

private void gvComplaint\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

Complaint cmp = (Complaint)gvComplaint.CurrentRow.DataBoundItem;

if (gvComplaint.Columns["Reply"].Index == e.ColumnIndex)

{

frmComplaintReply form = new frmComplaintReply(cmp);

form.Show();

}

}

}

public partial class frmCovid : Admin

{

public frmCovid()

{

InitializeComponent();

}

private void cmdGenerate\_Click(object sender, EventArgs e)

{

citizen person = citizenCRUD.isCitizenExist(txtToken.Text);

if (person != null)

{

frmCovidCard form = new frmCovidCard(person);

form.Show();

}

else

{

MessageBox.Show("Invalid!");

}

}

public partial class frmSehatApp : Form

{

public frmSehatApp()

{

InitializeComponent();

}

public void dataBind()

{

dataGridViewSahat.DataSource = null;

dataGridViewSahat.DataSource = citizenCRUD.SahatAppList;

dataGridViewSahat.Refresh();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

citizen person = (citizen)dataGridViewSahat.CurrentRow.DataBoundItem;

if (dataGridViewSahat.Columns["Accept"].Index == e.ColumnIndex)

{

//accpting and than delte from sahatapp list and add into selectedSahatList

citizenCRUD.addSahatSelectedCitizen(person);

citizenCRUD.deleteUserFromList(person,citizenCRUD.SahatAppList);

citizenCRUD.StoreIntoFileSahatApp(FilePath.sahatAppPath, FilePath.sahatSelectPath);

dataBind();

}

else if (dataGridViewSahat.Columns["Reject"].Index == e.ColumnIndex)

{

//delete application from sahatAppList

citizenCRUD.deleteUserFromList(person,citizenCRUD.SahatAppList);

citizenCRUD.StoreIntoFileSahatApp(FilePath.sahatAppPath, FilePath.sahatSelectPath);

dataBind();

}

else if (dataGridViewSahat.Columns["Inspect"].Index == e.ColumnIndex)

{

frmHealthInspect form = new frmHealthInspect();

Communication.Person = person;

form.Show();

}

}

private void frmSehatApp\_Load(object sender, EventArgs e)

{

citizenCRUD.sortedDatalist();

dataGridViewSahat.DataSource = citizenCRUD.SahatAppList;

}

private void citizenToolStripMenuItem1\_Click(object sender, EventArgs e)

{

frmSearchCitizen form = new frmSearchCitizen();

form.Show();

this.Hide();

}

private void sortedCitizenWithAgeToolStripMenuItem\_Click(object sender, EventArgs e)

{

frmSortedCitizen form = new frmSortedCitizen();

form.Show();

this.Hide();

}

private void graphicalStatisticsOfCityToolStripMenuItem\_Click(object sender, EventArgs e)

{

frmGraphicalStat form = new frmGraphicalStat();

form.Show();

this.Hide();

}

private void applicantRecordToolStripMenuItem\_Click(object sender, EventArgs e)

{

frmAddCitizen form = new frmAddCitizen();

form.Show();

this.Hide();

}

public partial class frmSehatApp : Form

{

public frmSehatApp()

{

InitializeComponent();

}

public void dataBind()

{

dataGridViewSahat.DataSource = null;

dataGridViewSahat.DataSource = citizenCRUD.SahatAppList;

dataGridViewSahat.Refresh();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

citizen person = (citizen)dataGridViewSahat.CurrentRow.DataBoundItem;

if (dataGridViewSahat.Columns["Accept"].Index == e.ColumnIndex)

{

//accpting and than delte from sahatapp list and add into selectedSahatList

citizenCRUD.addSahatSelectedCitizen(person);

citizenCRUD.deleteUserFromList(person,citizenCRUD.SahatAppList);

citizenCRUD.StoreIntoFileSahatApp(FilePath.sahatAppPath, FilePath.sahatSelectPath);

dataBind();

}

else if (dataGridViewSahat.Columns["Reject"].Index == e.ColumnIndex)

{

//delete application from sahatAppList

citizenCRUD.deleteUserFromList(person,citizenCRUD.SahatAppList);

citizenCRUD.StoreIntoFileSahatApp(FilePath.sahatAppPath, FilePath.sahatSelectPath);

dataBind();

}

else if (dataGridViewSahat.Columns["Inspect"].Index == e.ColumnIndex)

{

frmHealthInspect form = new frmHealthInspect();

Communication.Person = person;

form.Show();

}

}

private void frmSehatApp\_Load(object sender, EventArgs e)

{

citizenCRUD.sortedDatalist();

dataGridViewSahat.DataSource = citizenCRUD.SahatAppList;

}

private void citizenToolStripMenuItem1\_Click(object sender, EventArgs e)

{

frmSearchCitizen form = new frmSearchCitizen();

form.Show();

this.Hide();

}

private void sortedCitizenWithAgeToolStripMenuItem\_Click(object sender, EventArgs e)

{

frmSortedCitizen form = new frmSortedCitizen();

form.Show();

this.Hide();

}

MessageBox.Show("Your tacking id is "+person.TokenNumber);

else

MessageBox.Show("No tracing ID found..Your application is in process");

}

else

{

MessageBox.Show("No registeres application found against this Cnic");

}

}

private void cmdtrack\_Click(object sender, EventArgs e)

{

if (int.Parse(txtTrack.Text) < 2000 && int.Parse(txtTrack.Text) > 1000)

{

citizen person = citizenCRUD.citizenFromTokenNumber(int.Parse(txtTrack.Text));

if (person != null)

{

MessageBox.Show(person.Name +"your application against tacking id " + person.TokenNumber+"is in completion mode.");

}

else

{

MessageBox.Show("Invalid!");

}

}

else

{

MessageBox.Show("Invalid!");

}

}

private void cmdFind\_Click\_1(object sender, EventArgs e)

{

pnlFind1.Show();

pnlFind2.Show();

}

}

}

form.Show();

this.Hide();

}

}

else

{

MessageBox.Show("Invalid Credentials");

}

}

private void cmdClose\_Click(object sender, EventArgs e)

{

this.Close();

}

private void cmdReset\_Click(object sender, EventArgs e)

{

txtPassword.Text = "";

txtUserName.Text = "";

}

private void lnkSignUp\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

SignUp form = new SignUp();

form.Show();

this.Hide();

}

private void txtUserName\_MouseClick(object sender, MouseEventArgs e)

{

if(txtUserName.Text== "Username")

txtUserName.Clear();

}

private void txtPassword\_MouseClick(object sender, MouseEventArgs e)

{

if (txtPassword.Text == "Password")

txtPassword.Clear();

}

}

}