Final Project Documentation

“Payroll Management System”

A Documentation
Presented to the
Information Technology and Information Systems Department
Adamson University

In partial fulfillment
of the requirements for the course
Computer Programming 1
under the degree of
Bachelor of Science in Information Technology

By
<Arellano, Janine>
<Donato, Jhasmine Joy>
<Santos, Rovic>
<Valdez, Daniel>

< Mrs. Rizalina Valencia >
Professor

<May 28, 2022>

TABLE OF CONTENTS

Chapter 1
Introduction

The system was conceptualized in the idea of a payroll system, as we all know in the modern world technology is at the helm of everything. Corporate systems require technicalities and system managements that efficiently organize the data of their employees and what not, a payroll system aims to solve the technicalities of a manual payroll system, as it helps the employees grasp their data in an organized and real time manner. A Payroll system is implemented through different forms and procedures which would later be discussed through the screenshots of the documentation itself. Companies in particular in today's setting tend to implement systems for lower manpower consumption and digitized databases to curb the need for manual recordings, a Payroll system solves this notion.

We came up with the idea in suggestion by our Professor Rizalina Valencia, we then created the program based on our learnings and the teachings of our professors, we also outsourced information from youtube in grasping the topic at a more advanced light. As on our own we would lack the necessary information and substance to competently formulate a database implemented payroll system.

Project References

DATABASE CONNECTION REFERENCES:

https://www.youtube.com/watch?v=72POpCW5-zE&ab_channel=ProgrammingKnowledge

https://www.youtube.com/watch?v=1YEz85gFj2s&ab_channel=ProgrammingKnowledge

https://www.youtube.com/watch?v=u3RadefPNBU&ab_channel=ProgrammingKnowledge

SYSTEM REFERENCES

https://www.flaticon.com/search?word=home&color=gradient&order_by=4

Project Members

Content Manager: Janine Arellano – is in-charge of the work flow of the group, she is also responsible for the implementation of the ideas on the system such as the Ui and main concept.

Lead Designer / Debugger: Jhasmine Joy Donato - is in-charge of creating the main gist of the Ui and designs, she is also responsible for debugging the codes and errors that the lead programmer is unable to perform.

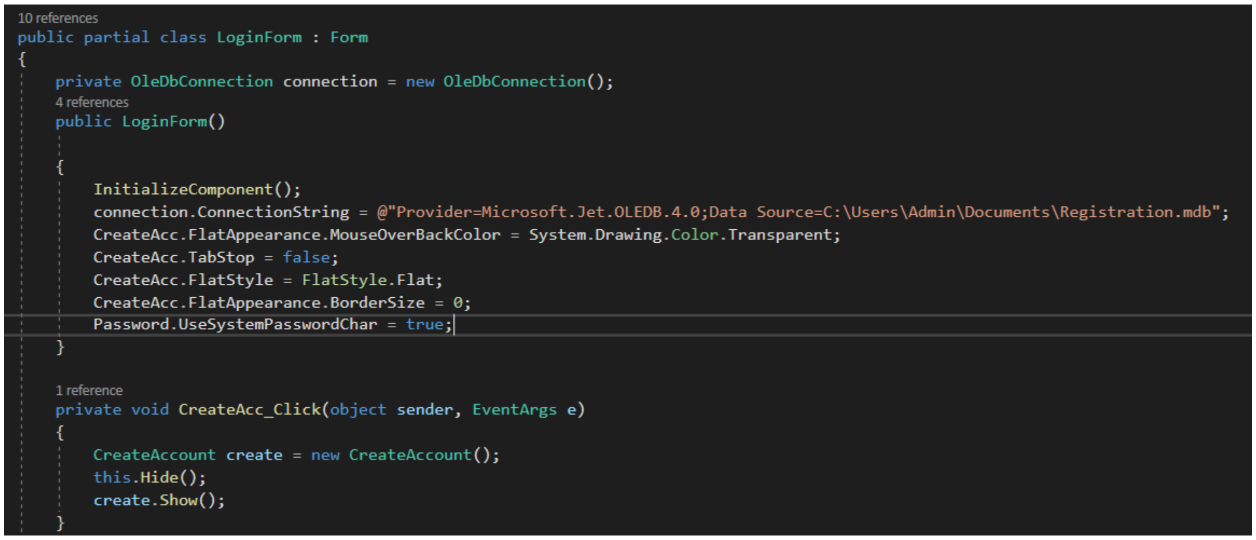
Lead Programmer: Daniel Valdez - is in-charge of the overall functions and coding of the system, he is also responsible for creating and connecting the database and the system.

Presenter / Documenter: Rovic Santos - is in-charge of the video presentation and recording of the demonstration, he is responsible for creating the documentation and paper presentation of the system.

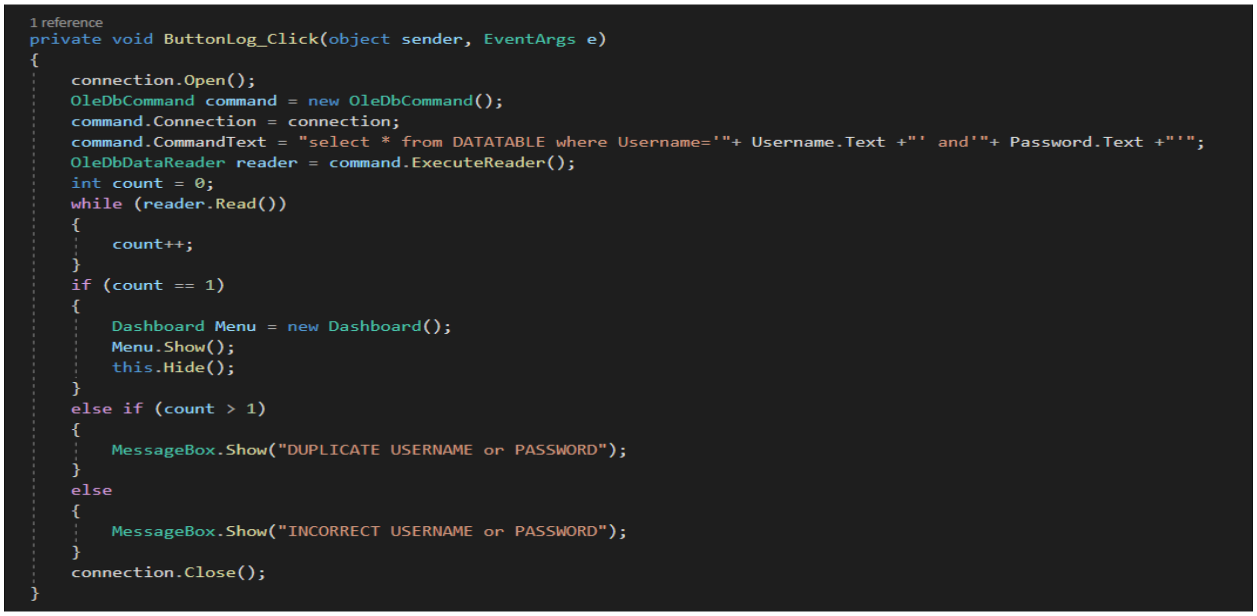
Chapter 2

System Framework / Codes

Login Form Code

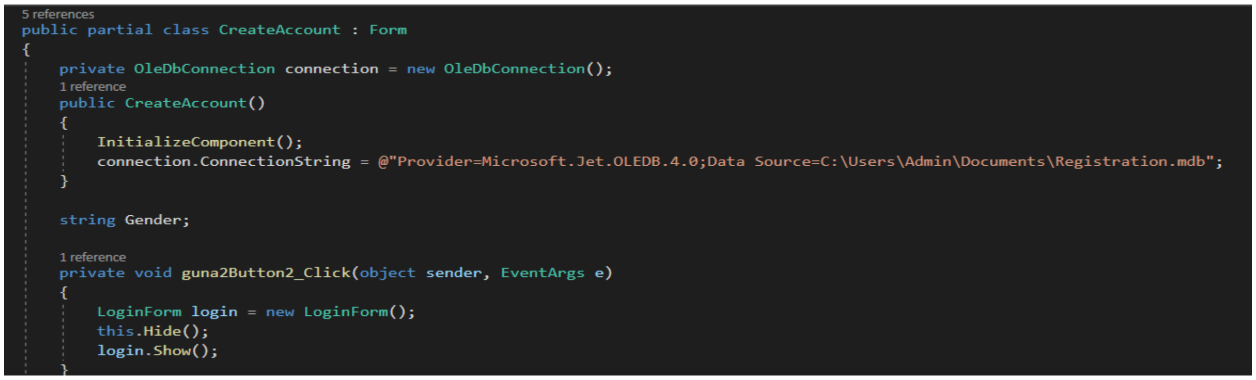


```
10 references
public partial class LoginForm : Form
{
    private OleDbConnection connection = new OleDbConnection();
    4 references
    public LoginForm()

    {
        InitializeComponent();
        connection.ConnectionString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Users\Admin\Documents\Registration.mdb";
        CreateAcc.FlatAppearance.MouseOverBackColor = System.Drawing.Color.Transparent;
        CreateAcc.TabStop = false;
        CreateAcc.FlatStyle = FlatStyle.Flat;
        CreateAcc.FlatAppearance.BorderSize = 0;
        Password.UseSystemPasswordChar = true;
    }

    1 reference
    private void CreateAcc_Click(object sender, EventArgs e)
    {
        CreateAccount create = new CreateAccount();
        this.Hide();
        create.Show();
    }
```

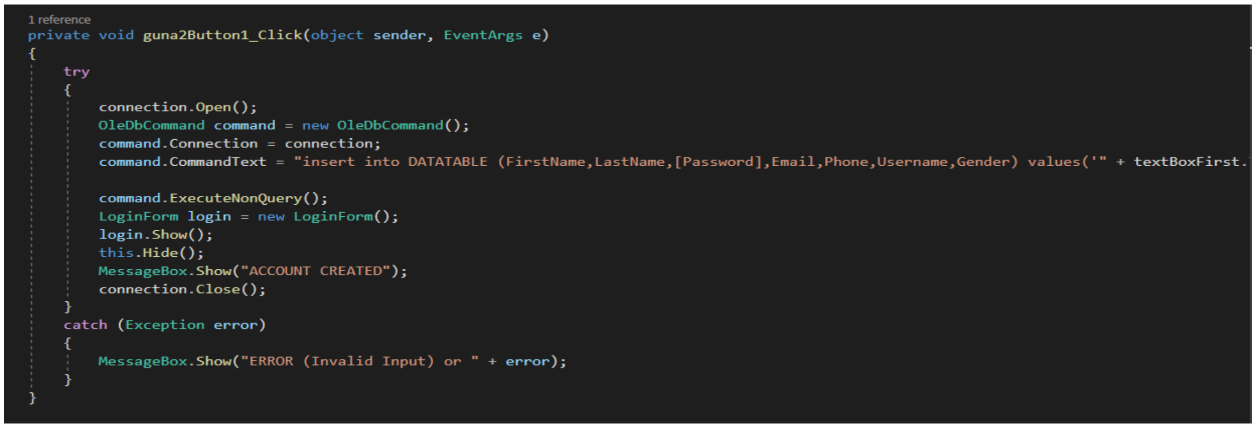


```
1 reference
private void ButtonLog_Click(object sender, EventArgs e)
{
    connection.Open();
    OleDbCommand command = new OleDbCommand();
    command.Connection = connection;
    command.CommandText = "select * from DATATABLE where Username='"+ Username.Text +"' and'"+ Password.Text +"'";
    OleDbDataReader reader = command.ExecuteReader();
    int count = 0;
    while (reader.Read())
    {
        count++;
    }
    if (count == 1)
    {
        Dashboard Menu = new Dashboard();
        Menu.Show();
        this.Hide();
    }
    else if (count > 1)
    {
        MessageBox.Show("DUPLICATE USERNAME or PASSWORD");
    }
    else
    {
        MessageBox.Show("INCORRECT USERNAME or PASSWORD");
    }
    connection.Close();
}
```

Registration Form Code:

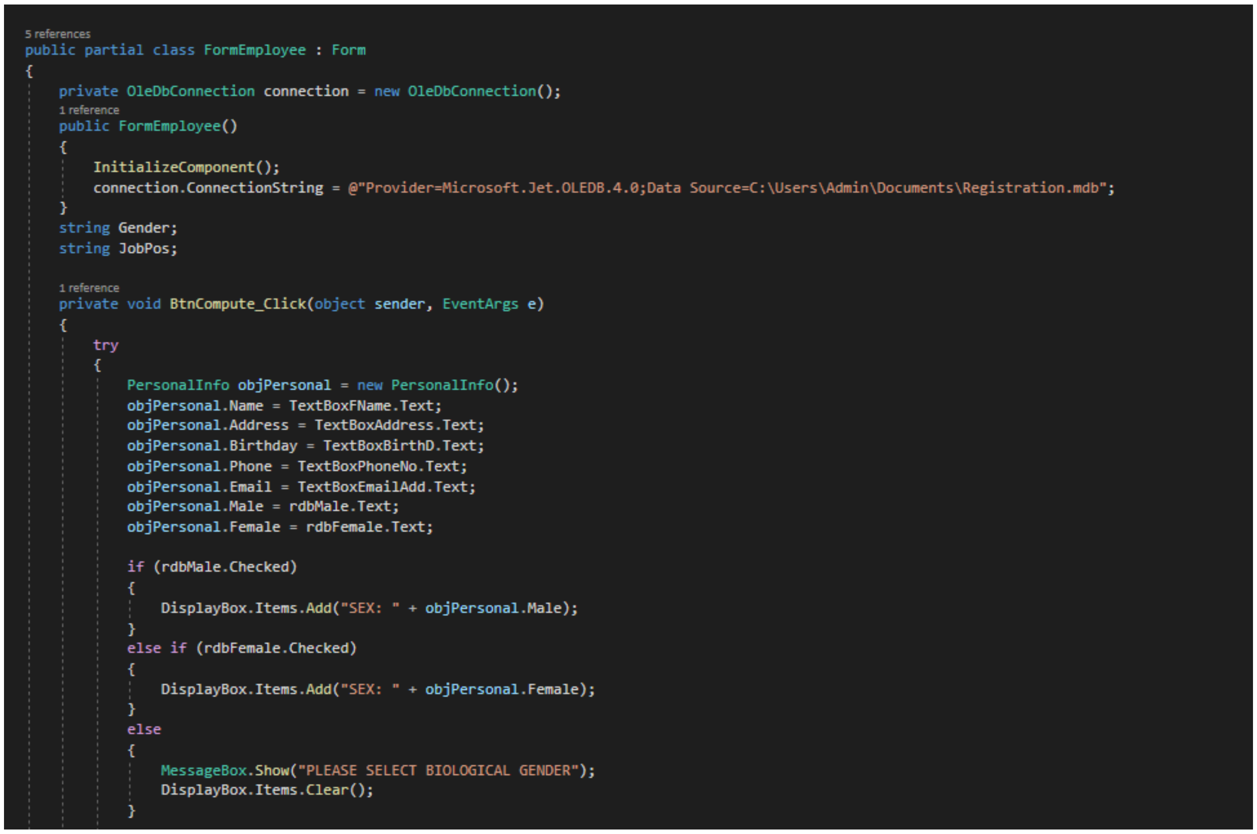


```
5 references
public partial class CreateAccount : Form
{
    private OleDbConnection connection = new OleDbConnection();
    1 reference
    public CreateAccount()
    {
        InitializeComponent();
        connection.ConnectionString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Users\Admin\Documents\Registration.mdb";
    }

    string Gender;

    1 reference
    private void guna2Button2_Click(object sender, EventArgs e)
    {
        LoginForm login = new LoginForm();
        this.Hide();
        login.Show();
    }
```

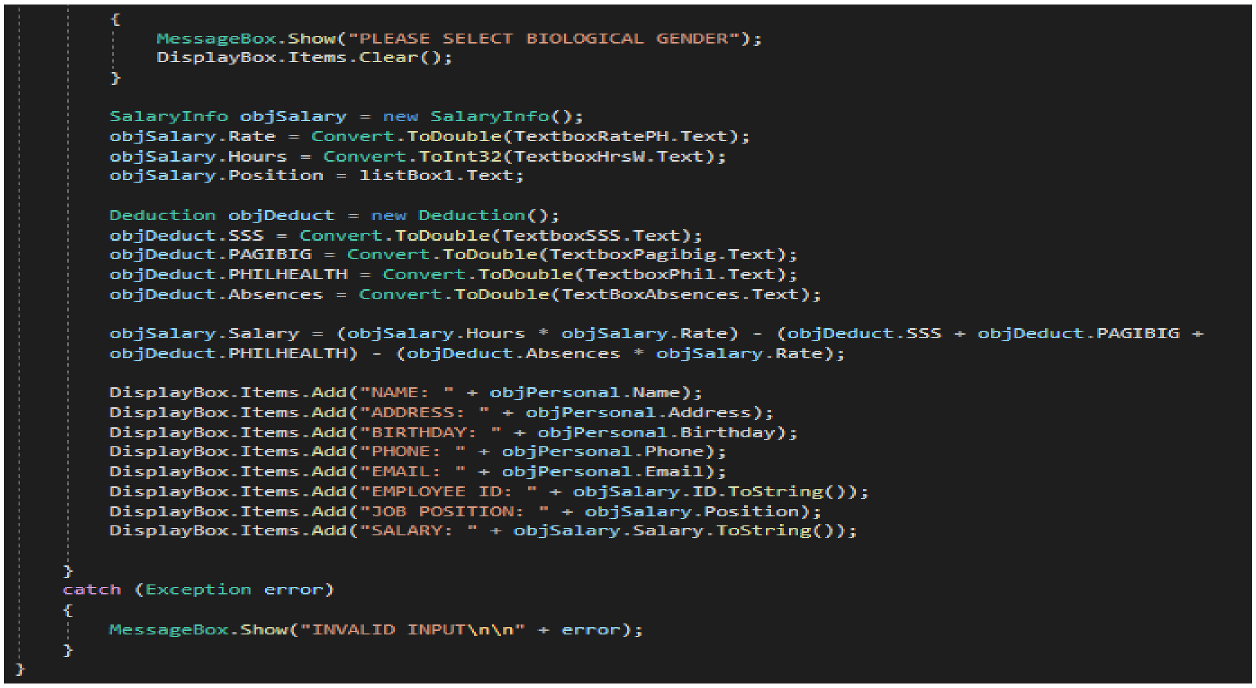


```
1 reference
private void guna2Button1_Click(object sender, EventArgs e)
{
    try
    {
        connection.Open();
        OleDbCommand command = new OleDbCommand();
        command.Connection = connection;
        command.CommandText = "insert into DATATABLE (FirstName,LastName,[Password],Email,Phone,Username,Gender) values('" + textBoxFirst.

        command.ExecuteNonQuery();
        LoginForm login = new LoginForm();
        login.Show();
        this.Hide();
        MessageBox.Show("ACCOUNT CREATED");
        connection.Close();
    }
    catch (Exception error)
    {
        MessageBox.Show("ERROR (Invalid Input) or " + error);
    }
}
```

Add Employee Form:

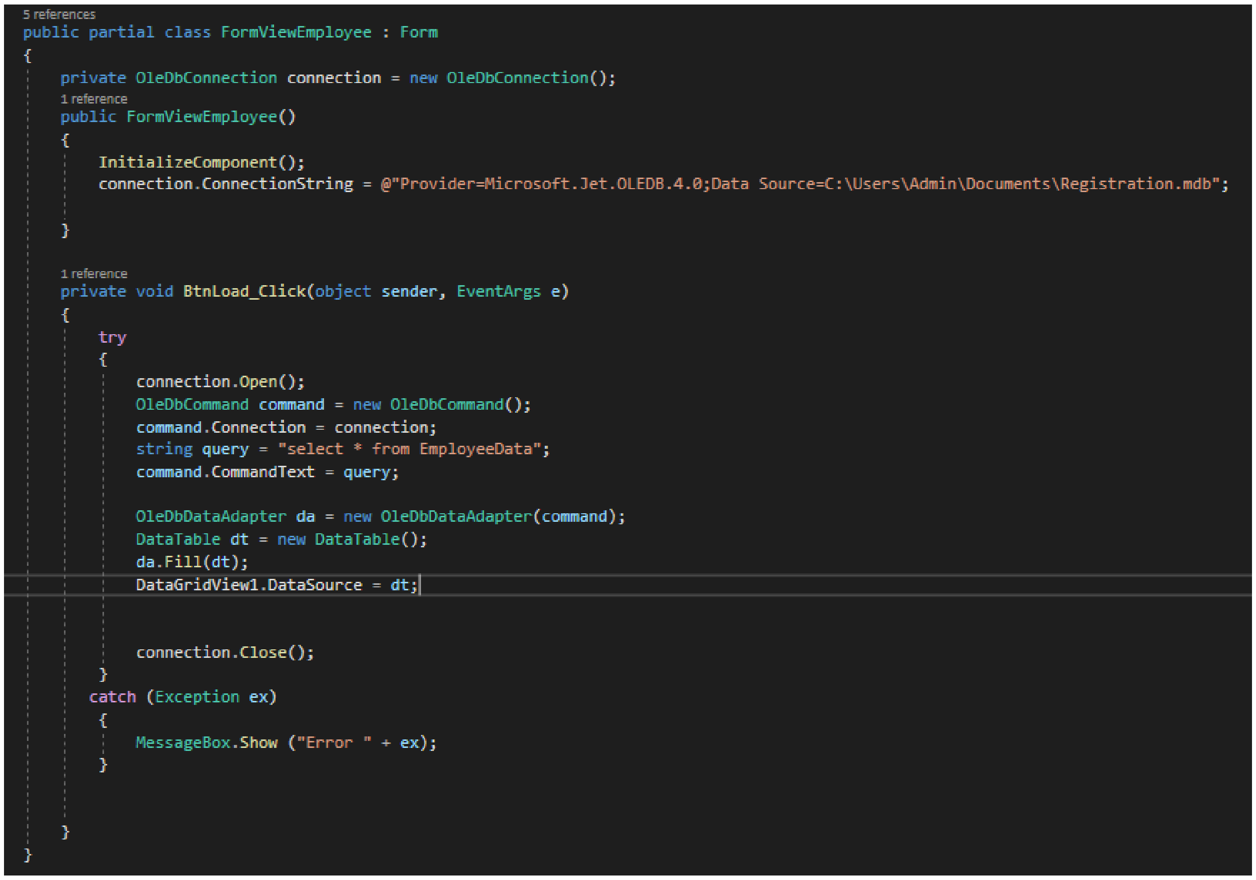


```
5 references
public partial class FormEmployee : Form
{
    private OleDbConnection connection = new OleDbConnection();
    1 reference
    public FormEmployee()
    {
        InitializeComponent();
        connection.ConnectionString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Users\Admin\Documents\Registration.mdb";
    }
    string Gender;
    string JobPos;

    1 reference
    private void BtnCompute_Click(object sender, EventArgs e)
    {
        try
        {
            PersonalInfo objPersonal = new PersonalInfo();
            objPersonal.Name = TextBoxFName.Text;
            objPersonal.Address = TextBoxAddress.Text;
            objPersonal.Birthday = TextBoxBirthD.Text;
            objPersonal.Phone = TextBoxPhoneNo.Text;
            objPersonal.Email = TextBoxEmailAdd.Text;
            objPersonal.Male = rdbMale.Text;
            objPersonal.Female = rdbFemale.Text;

            if (rdbMale.Checked)
            {
                DisplayBox.Items.Add("SEX: " + objPersonal.Male);
            }
            else if (rdbFemale.Checked)
            {
                DisplayBox.Items.Add("SEX: " + objPersonal.Female);
            }
            else
            {
                MessageBox.Show("PLEASE SELECT BIOLOGICAL GENDER");
                DisplayBox.Items.Clear();
            }
```



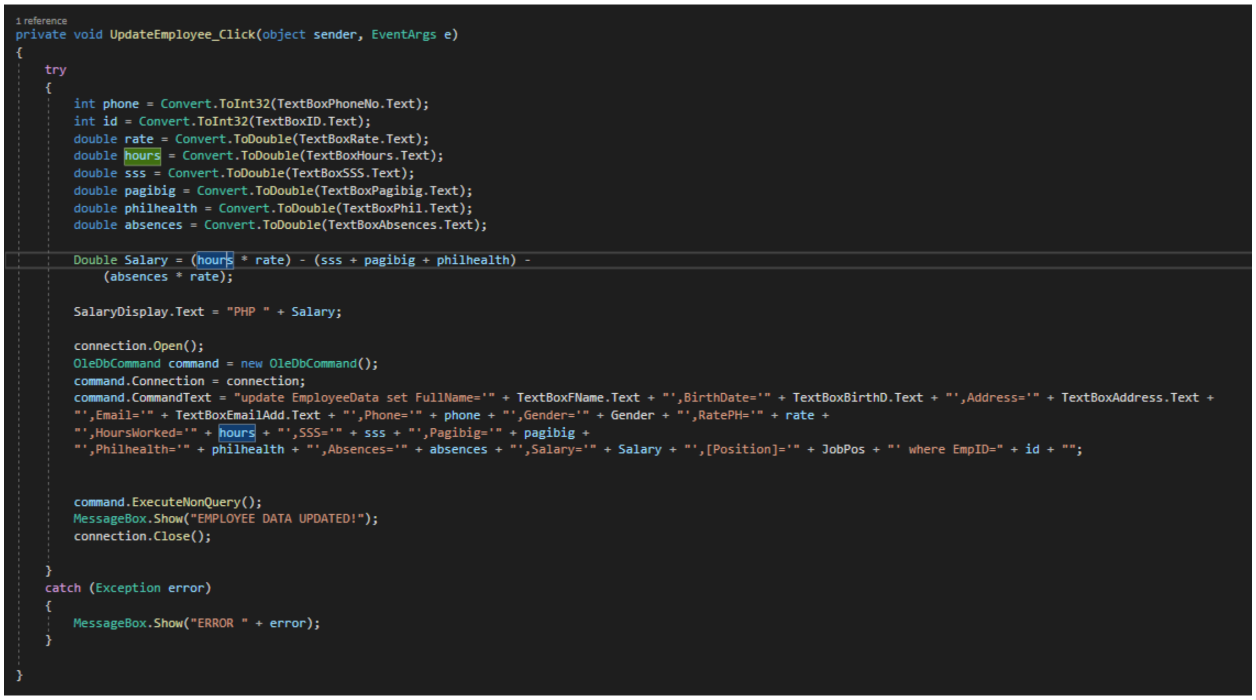
```
            {
                MessageBox.Show("PLEASE SELECT BIOLOGICAL GENDER");
                DisplayBox.Items.Clear();
            }

            SalaryInfo objSalary = new SalaryInfo();
            objSalary.Rate = Convert.ToDouble(TextboxRatePH.Text);
            objSalary.Hours = Convert.ToInt32(TextboxHrsW.Text);
            objSalary.Position = listBox1.Text;

            Deduction objDeduct = new Deduction();
            objDeduct.SSS = Convert.ToDouble(TextboxSSS.Text);
            objDeduct.PAGIBIG = Convert.ToDouble(TextboxPagibig.Text);
            objDeduct.PHILHEALTH = Convert.ToDouble(TextboxPhil.Text);
            objDeduct.Absences = Convert.ToDouble(TextBoxAbsences.Text);

            objSalary.Salary = (objSalary.Hours * objSalary.Rate) - (objDeduct.SSS + objDeduct.PAGIBIG +
            objDeduct.PHILHEALTH) - (objDeduct.Absences * objSalary.Rate);

            DisplayBox.Items.Add("NAME: " + objPersonal.Name);
            DisplayBox.Items.Add("ADDRESS: " + objPersonal.Address);
            DisplayBox.Items.Add("BIRTHDAY: " + objPersonal.Birthday);
            DisplayBox.Items.Add("PHONE: " + objPersonal.Phone);
            DisplayBox.Items.Add("EMAIL: " + objPersonal.Email);
            DisplayBox.Items.Add("EMPLOYEE ID: " + objSalary.ID.ToString());
            DisplayBox.Items.Add("JOB POSITION: " + objSalary.Position);
            DisplayBox.Items.Add("SALARY: " + objSalary.Salary.ToString());

        }
        catch (Exception error)
        {
            MessageBox.Show("INVALID INPUT\n\n" + error);
        }
    }
```

View Employee Form:

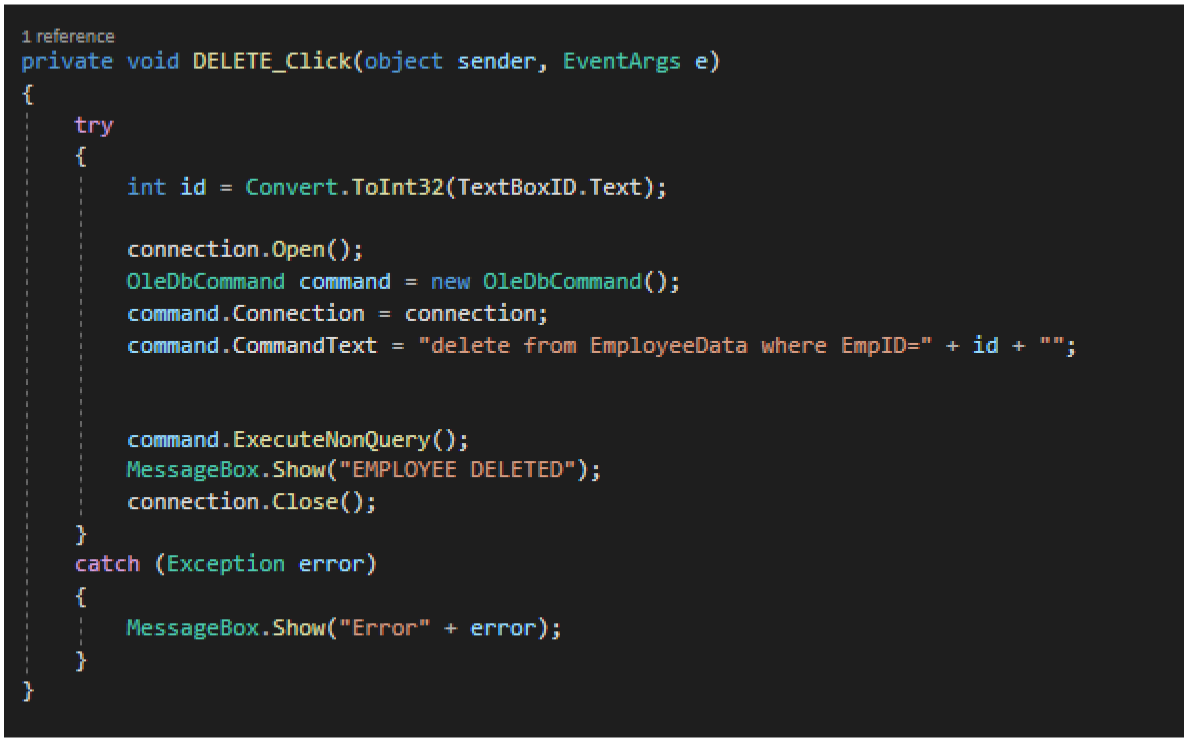


```
5 references
public partial class FormViewEmployee : Form
{
    private OleDbConnection connection = new OleDbConnection();
    1 reference
    public FormViewEmployee()
    {
        InitializeComponent();
        connection.ConnectionString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Users\Admin\Documents\Registration.mdb";

    }

    1 reference
    private void BtnLoad_Click(object sender, EventArgs e)
    {
        try
        {
            connection.Open();
            OleDbCommand command = new OleDbCommand();
            command.Connection = connection;
            string query = "select * from EmployeeData";
            command.CommandText = query;

            OleDbDataAdapter da = new OleDbDataAdapter(command);
            DataTable dt = new DataTable();
            da.Fill(dt);
            DataGridView1.DataSource = dt;


            connection.Close();
        }
      catch (Exception ex)
        {
            MessageBox.Show ("Error " + ex);
        }


    }
}
```

Update Employee & Delete Employee Form:

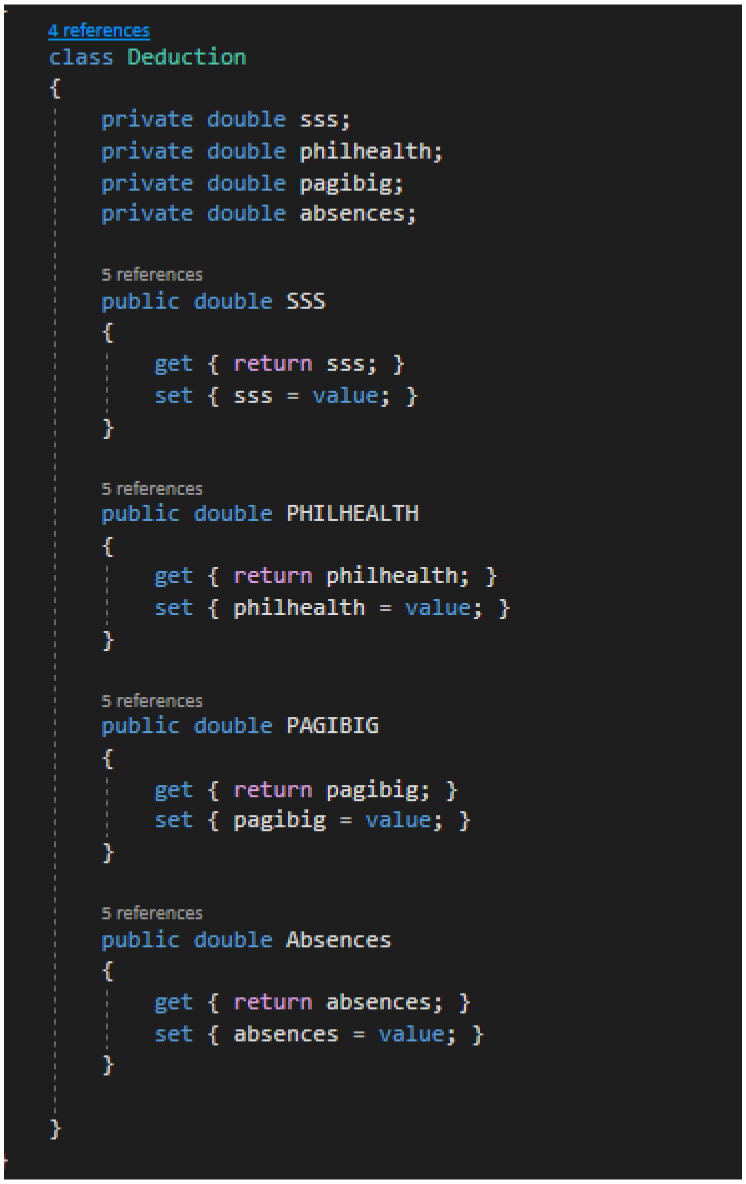


```
1 reference
private void UpdateEmployee_Click(object sender, EventArgs e)
{
    try
    {
        int phone = Convert.ToInt32(TextBoxPhoneNo.Text);
        int id = Convert.ToInt32(TextBoxID.Text);
        double rate = Convert.ToDouble(TextBoxRate.Text);
        double hours = Convert.ToDouble(TextBoxHours.Text);
        double sss = Convert.ToDouble(TextBoxSSS.Text);
        double pagibig = Convert.ToDouble(TextBoxPagibig.Text);
        double philhealth = Convert.ToDouble(TextBoxPhil.Text);
        double absences = Convert.ToDouble(TextBoxAbsences.Text);

        Double Salary = (hours * rate) - (sss + pagibig + philhealth) -
            (absences * rate);

        SalaryDisplay.Text = "PHP " + Salary;

        connection.Open();
        OleDbCommand command = new OleDbCommand();
        command.Connection = connection;
        command.CommandText = "update EmployeeData set FullName='" + TextBoxFName.Text + "',BirthDate='" + TextBoxBirthD.Text + "',Address='" + TextBoxAddress.Text +
        "',Email='" + TextBoxEmailAdd.Text + "',Phone='" + phone + "',Gender='" + Gender + "',RatePH='" + rate +
        "',HoursWorked='" + hours + "',SSS='" + sss + "',Pagibig='" + pagibig +
        "',Philhealth='" + philhealth + "',Absences='" + absences + "',Salary='" + Salary + "',[Position]='" + JobPos + "' where EmpID=" + id + "";


        command.ExecuteNonQuery();
        MessageBox.Show("EMPLOYEE DATA UPDATED!");
        connection.Close();

    }
    catch (Exception error)
    {
        MessageBox.Show("ERROR " + error);
    }

}
```



```
1 reference
private void DELETE_Click(object sender, EventArgs e)
{
    try
    {
        int id = Convert.ToInt32(TextBoxID.Text);

        connection.Open();
        OleDbCommand command = new OleDbCommand();
        command.Connection = connection;
        command.CommandText = "delete from EmployeeData where EmpID=" + id + "";


        command.ExecuteNonQuery();
        MessageBox.Show("EMPLOYEE DELETED");
        connection.Close();
    }
    catch (Exception error)
    {
        MessageBox.Show("Error" + error);
    }
}
```

Deductions:


```
4 references
class Deduction
{
    private double sss;
    private double philhealth;
    private double pagibig;
    private double absences;

    5 references
    public double SSS
    {
        get { return sss; }
        set { sss = value; }
    }

    5 references
    public double PHILHEALTH
    {
        get { return philhealth; }
        set { philhealth = value; }
    }

    5 references
    public double PAGIBIG
    {
        get { return pagibig; }
        set { pagibig = value; }
    }

    5 references
    public double Absences
    {
        get { return absences; }
        set { absences = value; }
    }

}
```

CODE EXPLANATIONS

LOG-IN FORM CODE:

The code above defines our log-in form's function. We connected it in our database to make our system dynamic and effective. The table name in MS Access database where the log-in form will retrieve the data that exists is called "DATATABLE" and that table stores the inputted data and information of the user upon registration. Catching errors on the other hand, we used try and catch syntax to ensure the program won't crash when the user inputters a wrong data type or required-to-fill-up fields. In addition, it will display in the message box if there are duplicated username and password.

REGISTRATION FORM CODE:

This form lets the user to add an account in order to proceed in log-in form and dashboard. It stores the inoutted values to the DATATABLE tablr in our database. And of course it comes with the try and catch syntax to avoid program crash.

ADD EMPLOYEE FORM:

This form lets the user add an employee by entering their complete details of the employee such as personal information and salary information. After then, it will display the summary of the values in the listbox tool. And by clicking the ADD EMPLOYEE button, the inputted values will be saved to the database record. The Employee ID of the user won't be duplicated since it is set as AUTONUMBER and as a PRIMARY KEY in the database.

VIEW EMPLOYEE CODE:

The code will function to let the user view the existing list of employees in the database table. We used the property OleDbAdapter to display the data in EmployeeData in a data grid view.

UPDATE & DELETE EMPLOYEE CODE:

Update Employee Code will let the user update the employee's specific details by filling all fields and clicking update. Take note that Employee ID is the most important one because it is the primary key. Employee ID acts as a unique identifier of the employee and can't be duplicated since it is set as AUTONUMBER in the database. Moving on with the delete employee code, the employer will just input the employee's ID and click the delete button to remove the record in the database.

CLASSES

PersonalInfo.cs Class:

Class used to create the Personal Information of the employee as an object.

SalaryInfo.cs

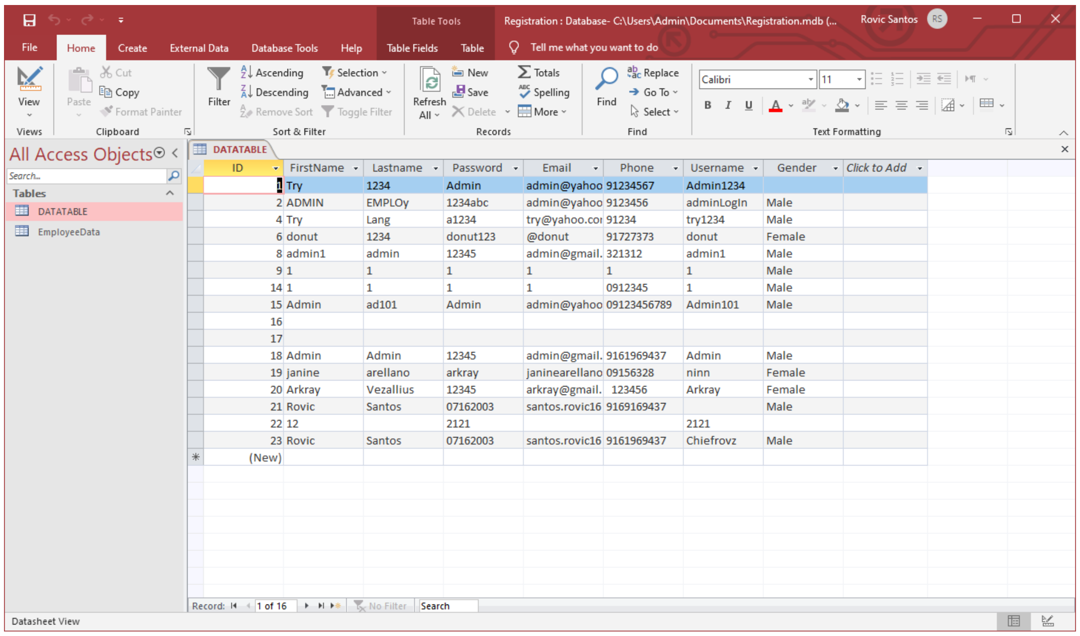
Class used to create the salary computation of the employee as an object.

Deduction.cs

Class used to calculate the deductions of the employee from their salary.

DATABASE

REGISTRATION & LOG-IN FORM DATABASE



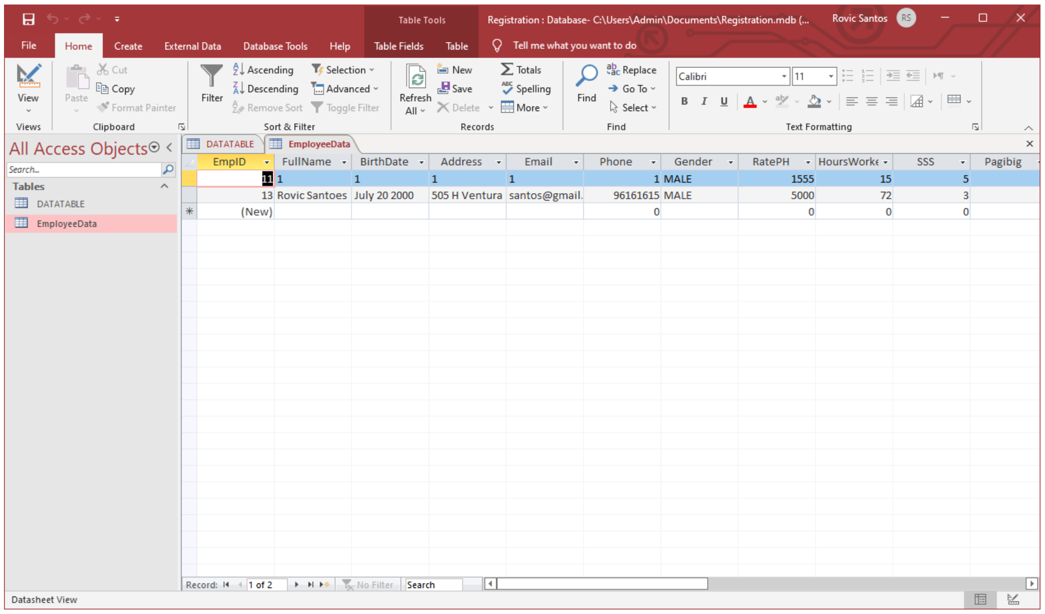
Registration : Database- C:\Users\Admin\Documents\Registration.mdb (...

DATATABLE

ID	FirstName	Lastname	Password	Email	Phone	Username	Gender	Click to Add
1	Try	1234	Admin	admin@yahoo	91234567	Admin1234		
2	ADMIN	EMPLOy	1234abc	admin@yahoo	9123456	adminLogIn	Male	
4	Try	Lang	a1234	try@yahoo.coı	91234	try1234	Male	
6	donut	1234	donut123	@donut	91727373	donut	Female	
8	admin1	admin	12345	admin@gmail.	321312	admin1	Male	
9	1	1	1	1	1	1	Male	
14	1	1	1	1	0912345	1	Male	
15	Admin	ad101	Admin	admin@yahoo	09123456789	Admin101	Male	
16								
17								
18	Admin	Admin	12345	admin@gmail.	9161969437	Admin	Male	
19	janine	arellano	arkray	janinearellano	09156328	ninn	Female	
20	Arkray	Vezallius	12345	arkray@gmail.	123456	Arkray	Female	
21	Rovic	Santos	07162003	santos.rovic16	9169169437		Male	
22	12		2121			2121		
23	Rovic	Santos	07162003	santos.rovic16	9161969437	Chiefrovz	Male	
(New)								

Record: 1 of 16

ADD EMPLOYEE, REMOVE EMPLOYEE, UPDATE & DELETE EMPLOYEE DATABASE



Registration : Database- C:\Users\Admin\Documents\Registration.mdb (...

EmployeeData

EmpID	FullName	BirthDate	Address	Email	Phone	Gender	RatePH	HoursWorke	SSS	Pagibig
11	1	1	1	1	1	MALE	1555	15	5	
13	Rovic Santoes	July 20 2000	505 H Ventura	santos@gmail.	96161615	MALE	5000	72	3	
(New)					0		0	0	0	

Record: 1 of 2

DATABASE FUNCTION

We connected the forms that require database function and connection by declaring the following:

Using.System.Data.OleDb;

It is a namespace to let the program use the database.

Private OleDbConnection connection = new OleDbConnection();

To connect it with the data source

connection.connectionString = @ "";

This also allow the connection of the project to a specific database source by providing the connection string by that specific database inside the double quotation.

OleDbCommand

This is a keyword to let you use the statement insert, delete, update rows in data tables.

OleDbReader

This provides a forward-only way of reading in data rows. This is used for reading or validating a data from the data source or data table.

connection.Open();

connection.Close();

These allows to open and close the connection between the project and the database. Those two are indispensable to each other otherwise you'll get an error of complication in connection.

OleDbAdapter

This provides a communication between data sets and OleDb data sources. This is used for retrieving and saving data.

CommandText

Now, this property is what we use when transferring data from your project to the database. This is where you input your variable name of columns in database and variable name of tools in your projects such as text boxes and radiobuttons to transfer their values to the database using CommandText.

ExecuteNonQuery();

This one used for queries that doesn't return any data. For better comprehension, it is used in statements like insert, update, and delete.

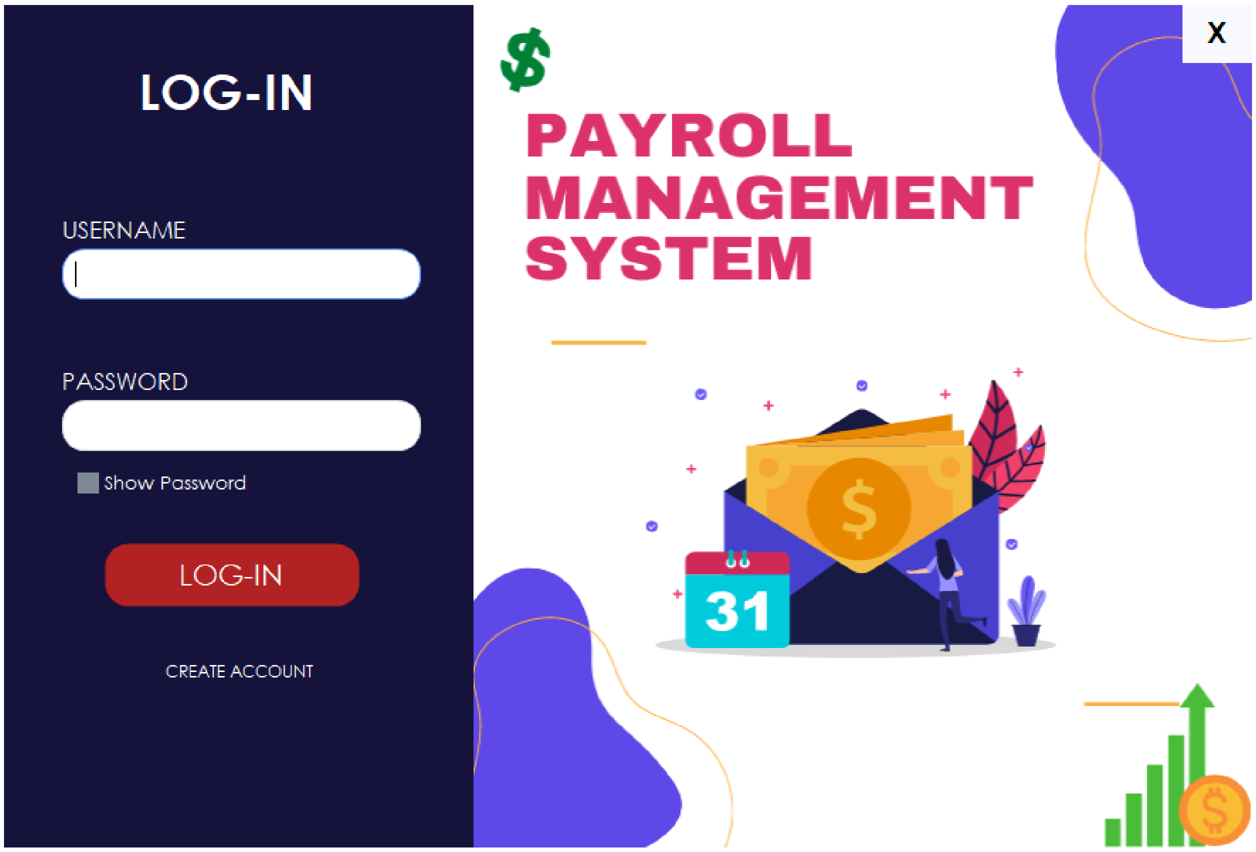
DataTable();

This property is used to define and store data. We used this property in viewing the added employees on our system along with Fill(); and OleDbAdapter(); syntax in DataGridView object to be able to view the data from database through data grid view.

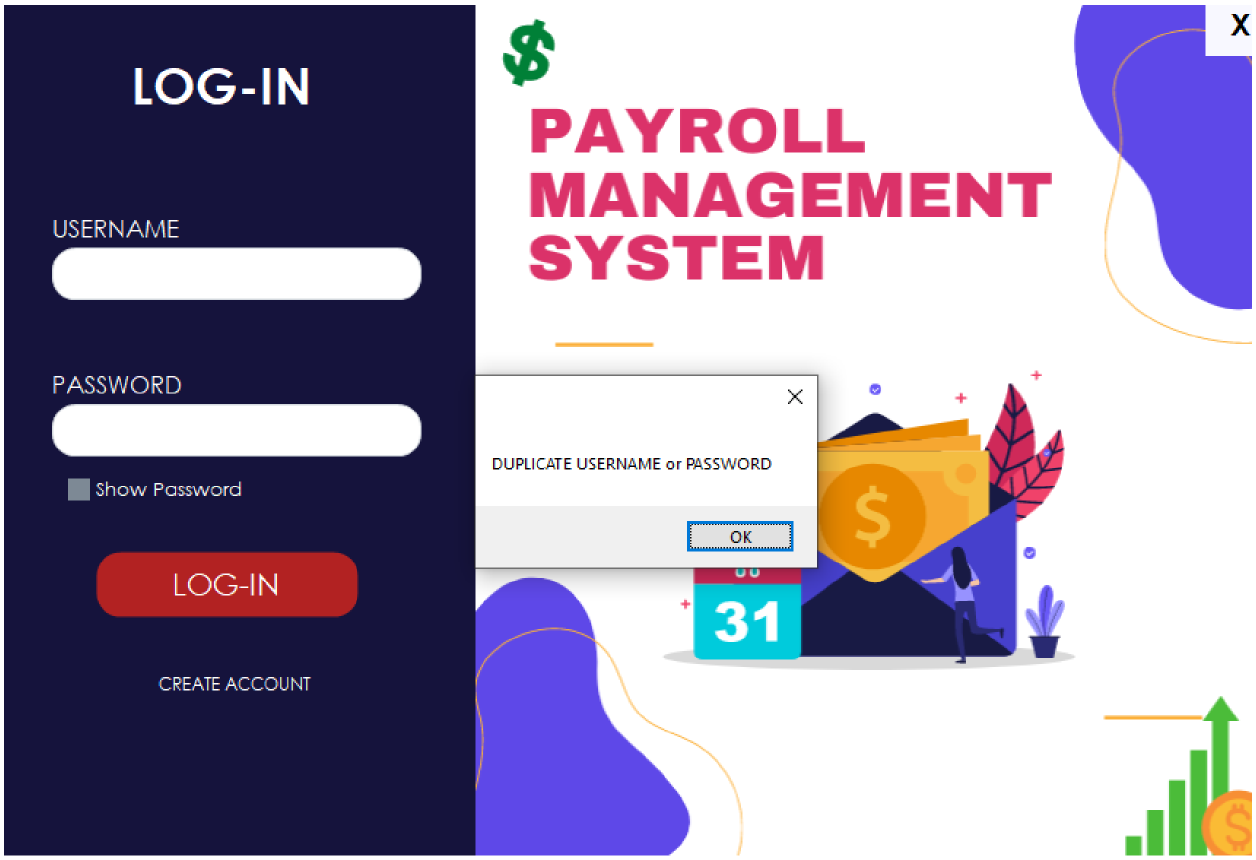
Chapter 3

Implementing the System

Login Form / Registration Form:

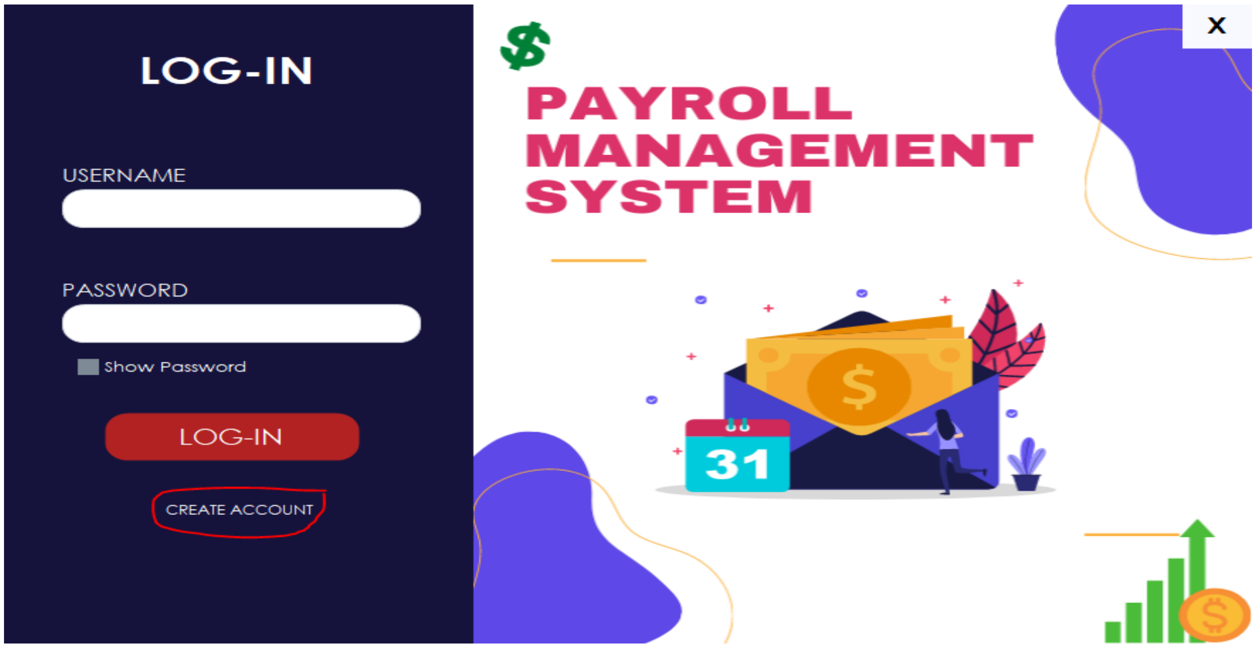


Here is the login form, upon filling the user will be able to proceed to the dashboard, nonetheless the user must first register if he/she has yet to be registered to the system. If he decides to enter without correct credentials a pop message will appear. As shown below.

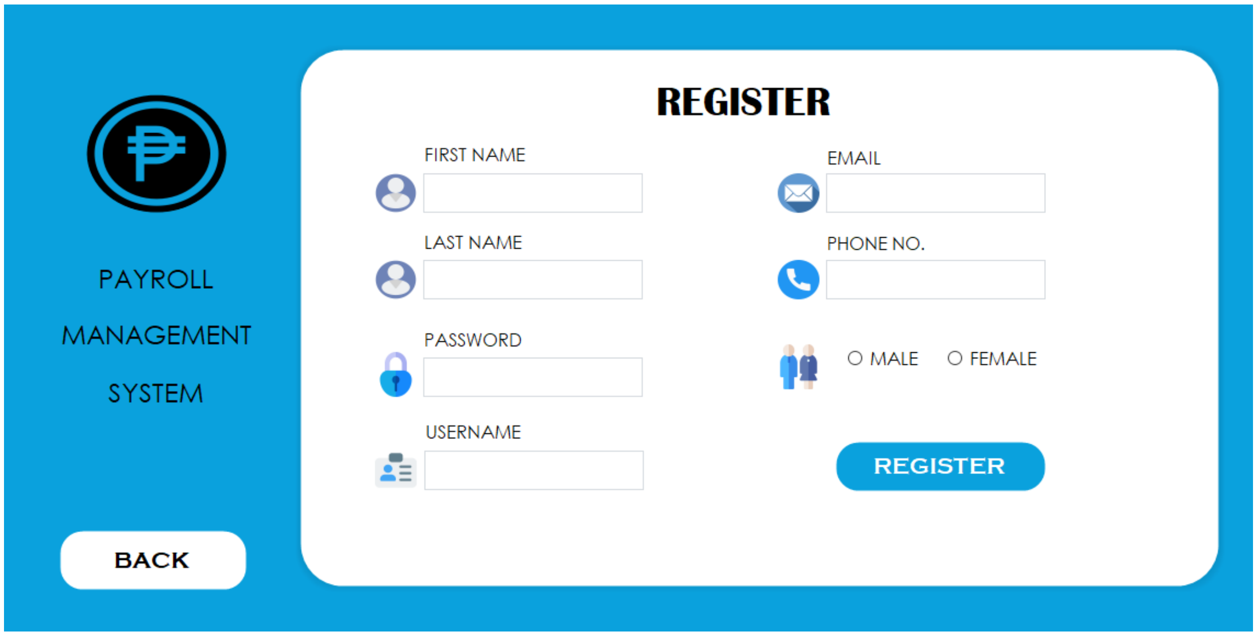


DUPLICATE USERNAME or PASSWORD

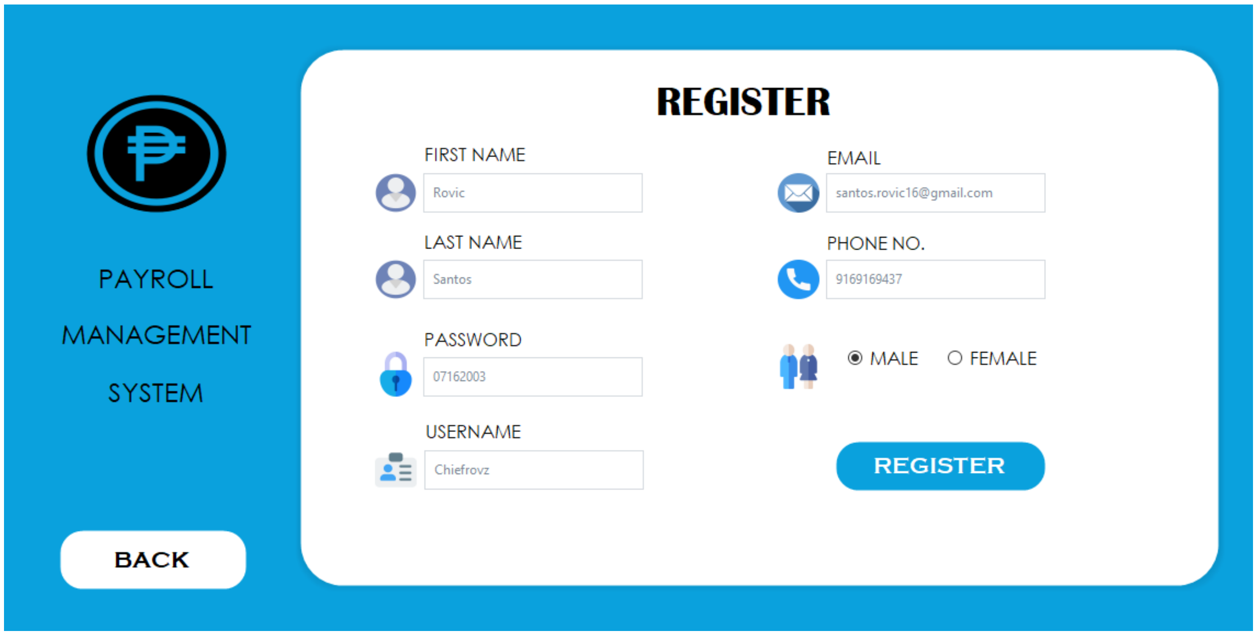
Now that the user decides to create an account they must press "CREATE ACCOUNT" Which is encircled red on the image shown below.



Upon pressing the "CREATE ACCOUNT" the user will now be directed to a form which they can use to input their data to our system's database.

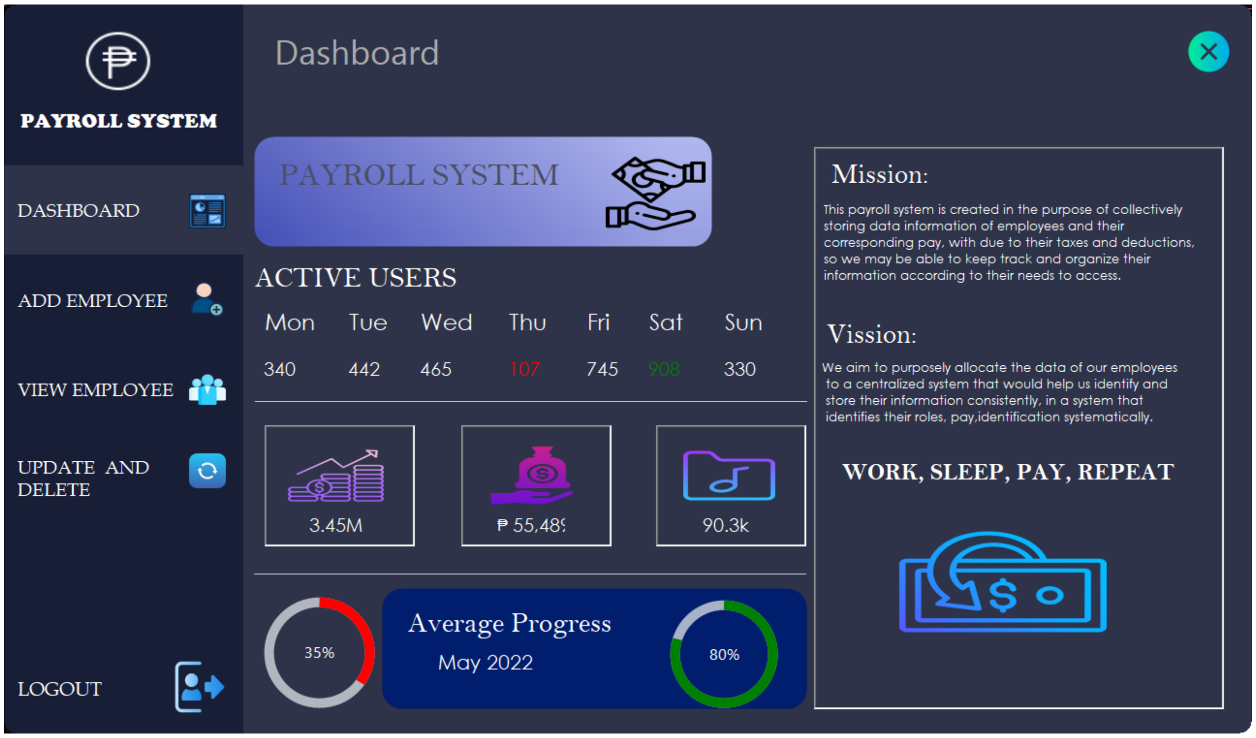


Here is a blank form shown above, the labels are self explanatory for the user. Now below is a filled form.



FIRST NAME	Rovic
LAST NAME	Santos
PASSWORD	07162003
USERNAME	Chiefrovz
EMAIL	santos.rovic16@gmail.com
PHONE NO.	9169169437
	[✓ MALE] FEMALE

Upon pressing the register, it automatically returns to the login form, from where a Messagebox appears and informs that he/she has successfully registered. Now the user may now press "LOGIN" which they would then be redirected to the "DASHBOARD". As shown by the image below.



ACTIVE USERS

Mon	Tue	Wed	Thu	Fri	Sat	Sun
340	442	465	107	745	908	330

Mission:

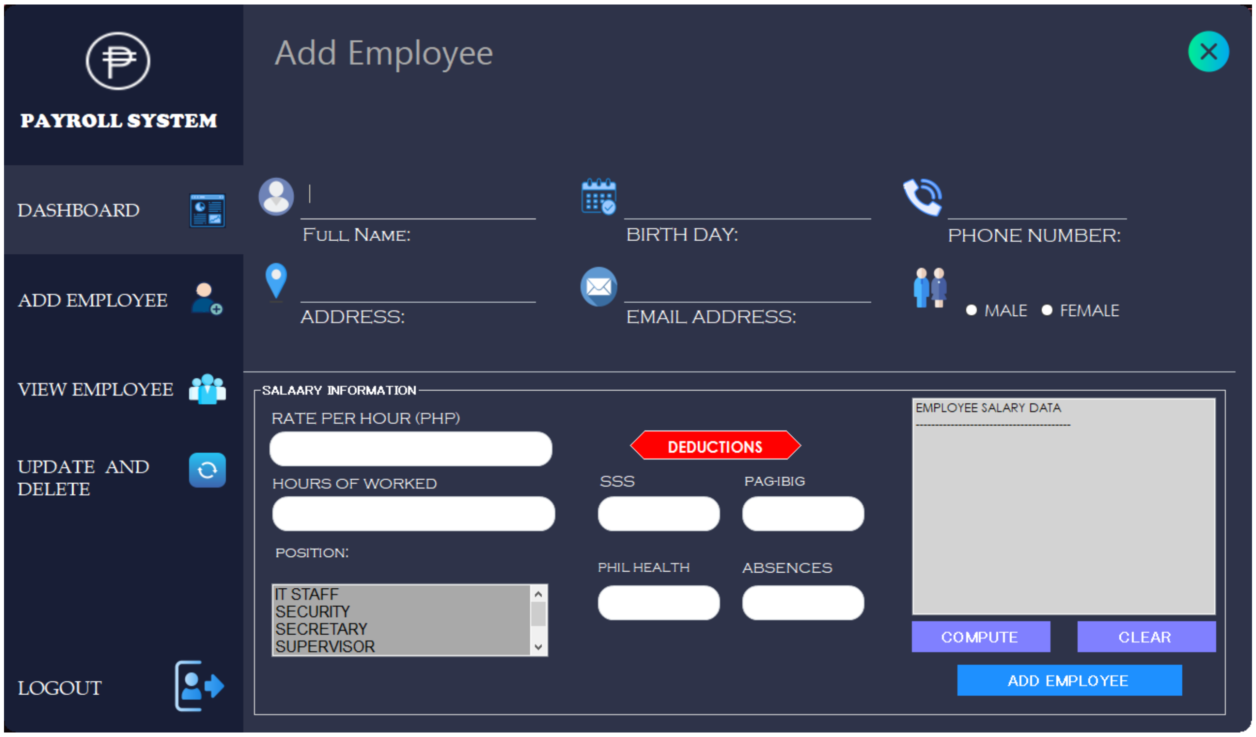
This payroll system is created in the purpose of collectively storing data information of employees and their corresponding pay, with due to their taxes and deductions, so we may be able to keep track and organize their information according to their needs to access.

Vission:

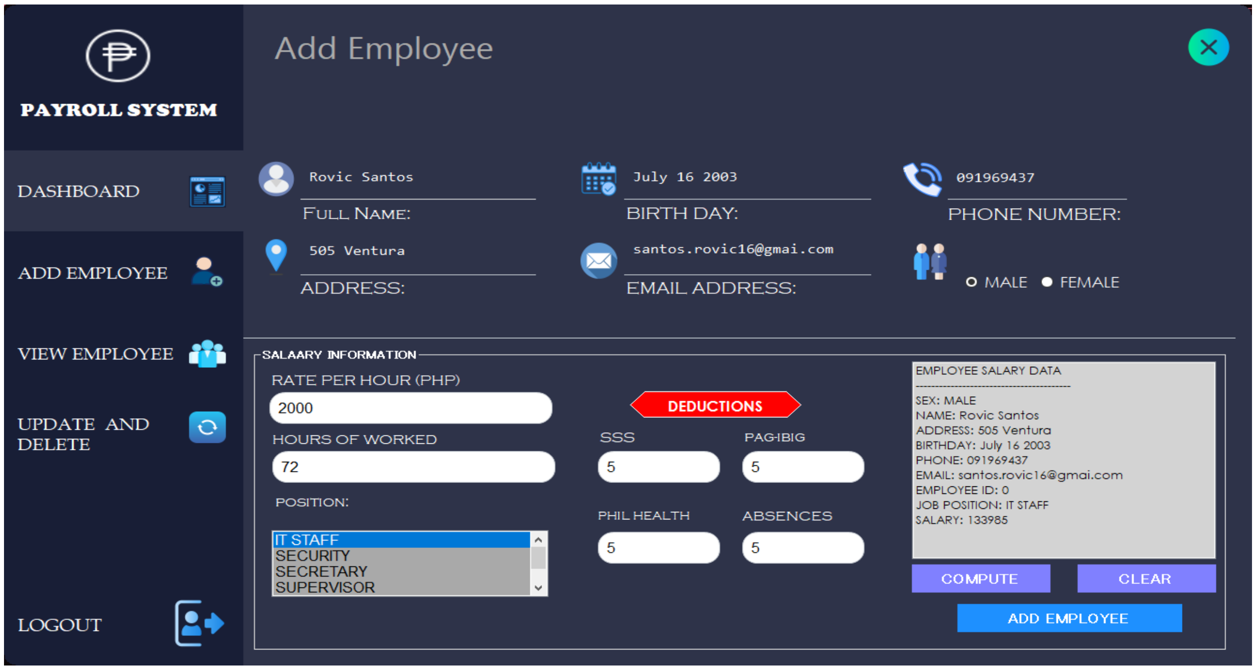
We aim to purposely allocate the data of our employees to a centralized system that would help us identify and store their information consistently, in a system that identifies their roles, pay,identification systematically.

WORK, SLEEP, PAY, REPEAT

From here the user may now access the features of the system, upon navigating they may proceed by pressing "ADD EMPLOYEE" where they will be directed to a form. Show below.



Now that the user has access to “ADD EMPLOYEE” they may now enter their credentials that are required by the form. As shown by the image below the user’s credentials upon entering and completing through “COMPUTE” their information is reflected upon the BOX.

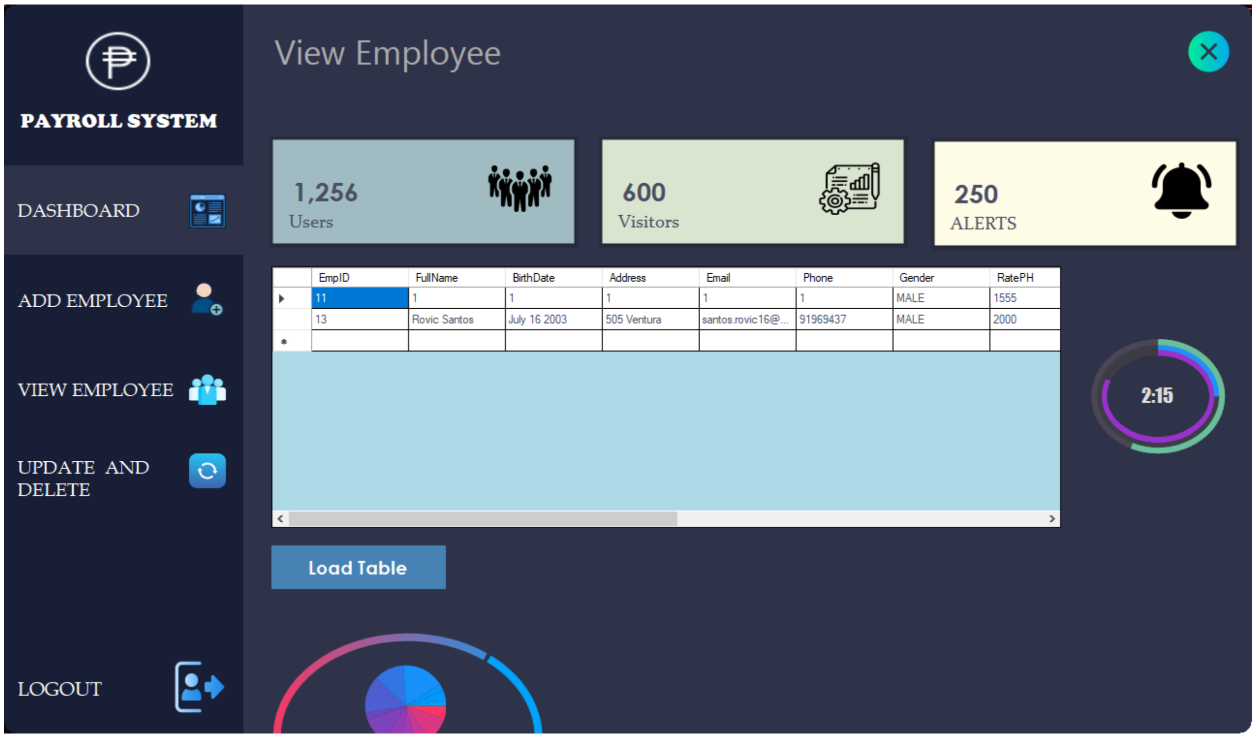


FULL NAME: Rovic Santos
BIRTH DAY: July 16 2003
PHONE NUMBER: 091969437
ADDRESS: 505 Ventura
EMAIL ADDRESS: santos.rovic16@gmai.com
[✓ MALE] FEMALE

SALAARY INFORMATION
RATE PER HOUR (PHP): 2000
HOURS OF WORKED: 72
POSITION: [✓ IT STAFF] SECURITY SECRETARY SUPERVISOR
SSS: 5
PAG-IBIG: 5
PHIL HEALTH: 5
ABSENCES: 5

```
EMPLOYEE SALARY DATA
--------------------------------------------
SEX: MALE
NAME: Rovic Santos
ADDRESS: 505 Ventura
BIRTHDAY: July 16 2003
PHONE: 091969437
EMAIL: santos.rovic16@gmai.com
EMPLOYEE ID: 0
JOB POSITION: IT STAFF
SALARY: 133985
```

As shown above the user's credentials are now verified upon pressing "ADD EMPLOYEE" the user's data will be routed to the database. Shown below.

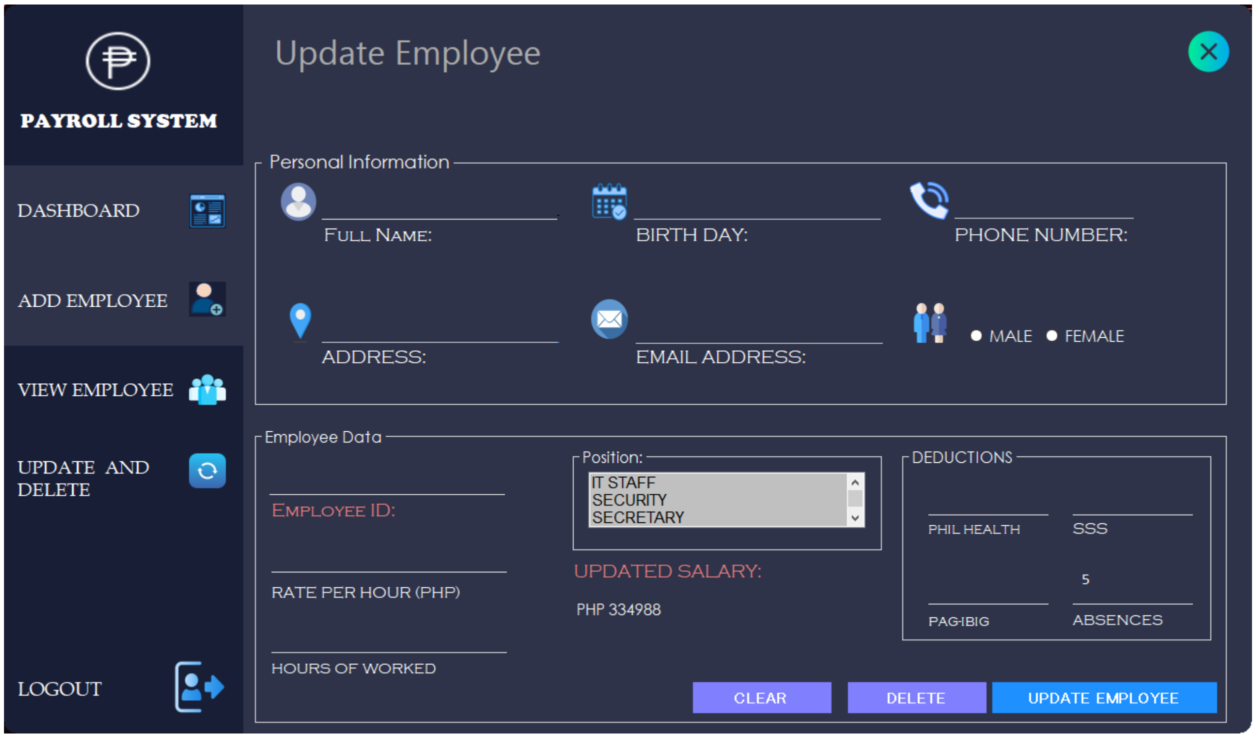


EmpID	FullName	BirthDate	Address	Email	Phone	Gender	RatePH
11	1	1	1	1	1	MALE	1555
13	Rovic Santos	July 16 2003	505 Ventura	santos.rovic16@...	91969437	MALE	2000

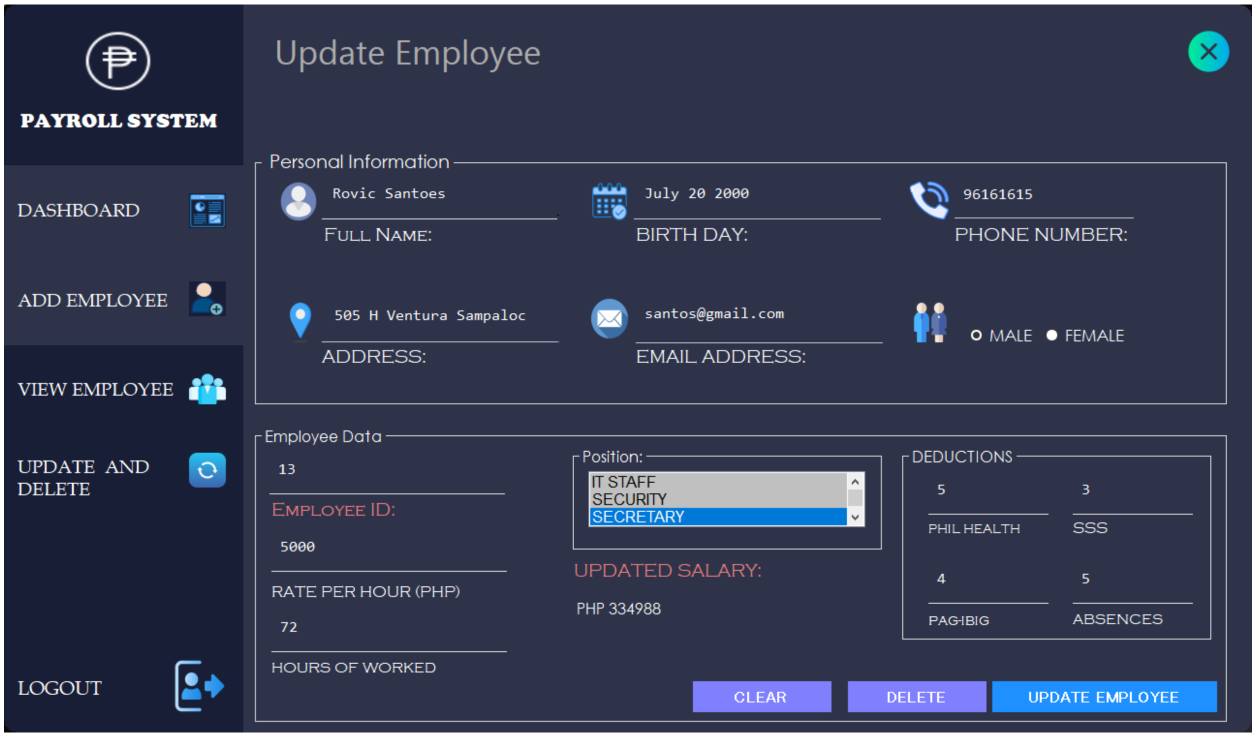
The Form in the image above is accessible through clicking "VIEW EMPLOYEE" which would show the real-time accounts that have registered. The data will be reflected real-time upon pressing "LOAD TABLE".

Now that the user is made aware of his credentials, if he wishes to make changes to his information he may extract his Employee ID from the DATA TABLE as shown above. Now he may proceed to the "UPDATE and DELETE" from where he could make changes to his account.

Lastly, shown below is the UPDATE and DELETE" It is similar to "ADD EMPLOYEE" as it follow the same principle albeit for modifying credentials or delete data.



Now that the user decides to fill it up, making changes to his user account by pressing "UPDATE EMPLOYEE" from where it automatically reflects back to the database and applies the changes in real time. Show below.



Update Employee

Personal Information

Rovic Santoes	July 20 2000	96161615
FULL NAME:	BIRTH DAY:	PHONE NUMBER:
505 H Ventura Sampaloc	santos@gmail.com	○ MALE ● FEMALE
ADDRESS:	EMAIL ADDRESS:	

Employee Data

13
EMPLOYEE ID:
5000
RATE PER HOUR (PHP)
72
HOURS OF WORKED

Position:
IT STAFF
SECURITY
SECRETARY

UPDATED SALARY:
PHP 334988

DEDUCTIONS

5	3
PHIL HEALTH	SSS
4	5
PAG-IBIG	ABSENCES

Now all of that is out of the way, that is all for the system implementation.

USER ACCESS

- The access level of the system is only exclusive to the administrator of the company which means the employer and the higher office can only have access on the system since it is the management of their employee's payroll and only the higher office can use the program.