

# Swift Payroll

## Project Process & Project Documentation

Project Name: Swift Payroll

Date: December 12, 2022

Year & Section: 2<sup>nd</sup> Year CS301

Group: 2

## Table of Contents

I. Project Charter.....	1
A. Introduction .....	1
B. Project Goals .....	1
C. Project Roles and Responsibilities.....	2
D. Scope Definition .....	3
E. Project Milestones .....	4
F. Project Plan.....	5
G. Related Documents.....	5
II. Design.....	6
A. Use-Case Diagram.....	6
B. User Interface .....	6
1. Main Form Home.....	6
2. About Page.....	7
3. The Team Page.....	7
4. Sign In Page .....	7
5. Sign Up Page.....	8
6. Forgot Password Page .....	8
7. Loading Screen .....	8
8. Dashboard (Admin).....	9
9. Dashboard (Human Resources Manager).....	9
10. Dashboard (Regular Employee) .....	10
11. Pay slip PDF.....	11
III. Software Specification.....	11
A. NuGet Packages and APIs .....	11
1. Guna.UI2.WinForms.....	11
2. MailKit & MimeKit .....	12
3. ITextSharp.....	12
4. System.Data.SQLite .....	12
IV. Software Implementation.....	13
A. EmployeeInfo Class.....	13
B. Sign In Page.....	17
C. Sign Up Page .....	18
D. Forgot Password Page .....	19
E. One-Time-Password Event.....	20
F. Pay slip calculation and generate PDF .....	21
G. How to run the Project.....	24

## I. Project Charter

### A. Introduction

The purpose of Swift Payroll System is mainly to provide automation to the company's human resources department. The main feature of this system where the Human Resources Manager can calculate the amount an employee owes based on factors such as the days they worked, their hourly wages or salaries, and whether they took overtime during the pay period. The system automatically adjusts net pay by calculating and subtracting SSS, Pag-IBIG, PhilHealth, Absences, and Tax. This system was designed for convenience, less time cost.

### B. Project Goals

1. User Friendly Design.
2. To compute the gross pay and net pay.
3. Manage Employee's Details.
4. To generate payroll and pay slips (PDF).
5. Login, Create Account, and Password Recovery User Controls.
6. Sending One time Password or OTP via Gmail for Account Recovery

### C. Project Roles and Responsibilities

The group had a meeting to decide the roles and responsibilities of the members. As a result, the group decided to assign the following:

1. **Timy Villarmia** as our group Leader. He has enough knowledge to help and guide each member on their given tasks. His responsibilities are to overlook the progress of the project. He is also tasked to deliver the proposal of the group's project.
2. **John Keith Bicare** was the one assigned to do the task of a programmer. He is tasked to code the program or functionality of the system though with minimal knowledge, Timy will guide and help in the process.
3. **Shinlee Armamento** was the one assigned to do the task of a designer. He is tasked to design the User Interface of the system. As per the design, it will still fall on the group's decision of what would be the final outcome of the design.
4. **Ginelyn Lucena** was the one assigned to do the task of the charter. She is tasked to update and make the project charter that has the information about the project.

## D. Scope Definition

**Swift Payroll** is a software that uses to pay its employees. The system includes calculating wages, the deduction of the assets and taxes the gross income, and generating pay slip for HR

### **The project will include:**

- ❖ The admins can manage the data from the database including the username and password
- ❖ The HR can update the employee's details such as: Job Title, Job Type, and Department.
- ❖ Profile User Control
  - Manage own profile
- ❖ Login User Control
- ❖ Account Recovery User Control
  - Email must be in the database to send an OTP to the email via Gmail
  - Change Password
- ❖ Create Account User Control
- ❖ Generating Pay slip for HR and saving it as PDF.
- ❖ Payroll User Control
  - For HR, Manage employee's pay slip.
  - For Regular Employee, View recent and past pay slip.

### **The project will not include:**

- ❖ Real-time tracking of Time IN/OUT of an employee.
- ❖ Real-time tracking of Absences and Lateness of an employee.
- ❖ Performance Report Dashboard
- ❖ Leave request and vacation request

E. Project Milestones

***Project Approval***

---

*Instructor Signature*

***Requirements Review***

---

*Instructor Signature*

***Design Approval***

---

*Instructor Signature*

***Project phase milestones***

---

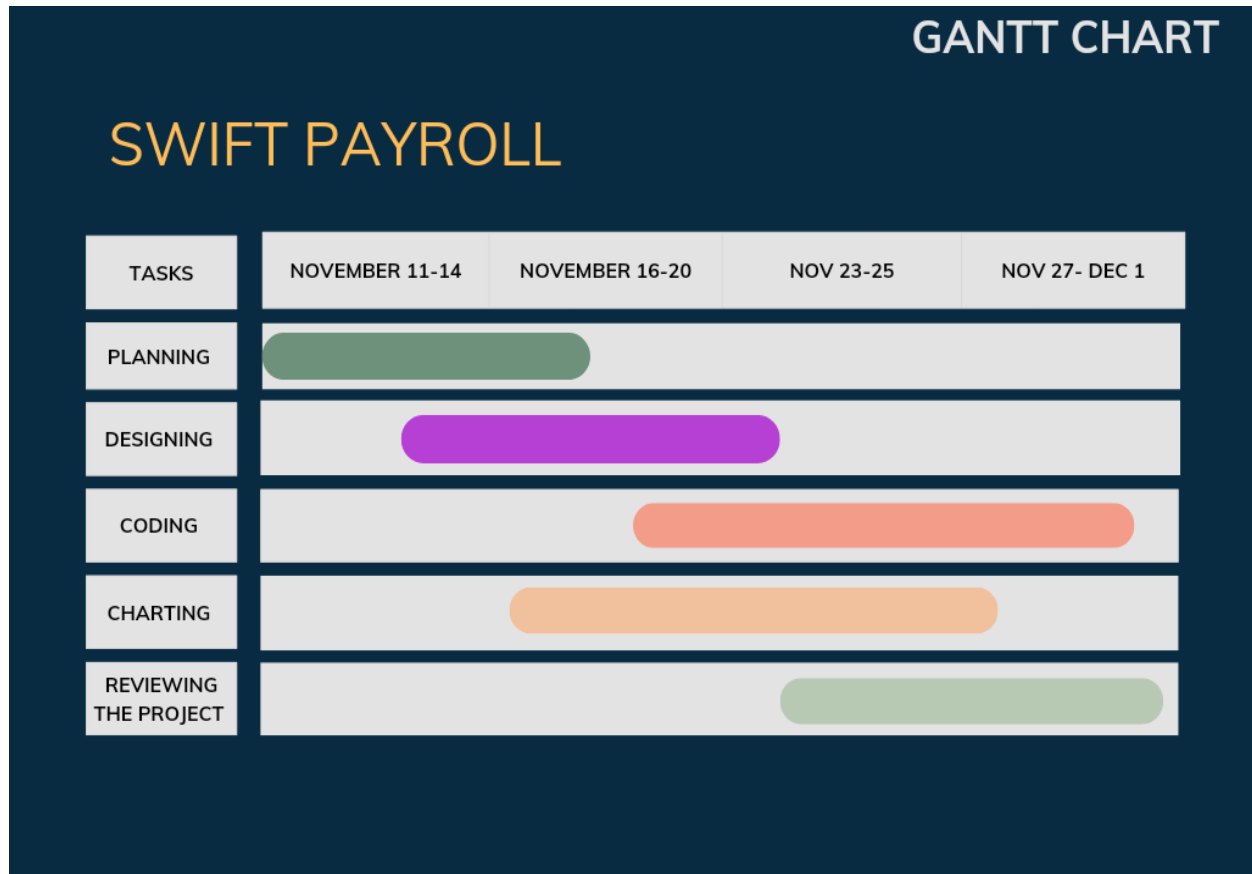
*Instructor Signature*

***Final Approval***

---

*Instructor Signature*

## F. Project Plan



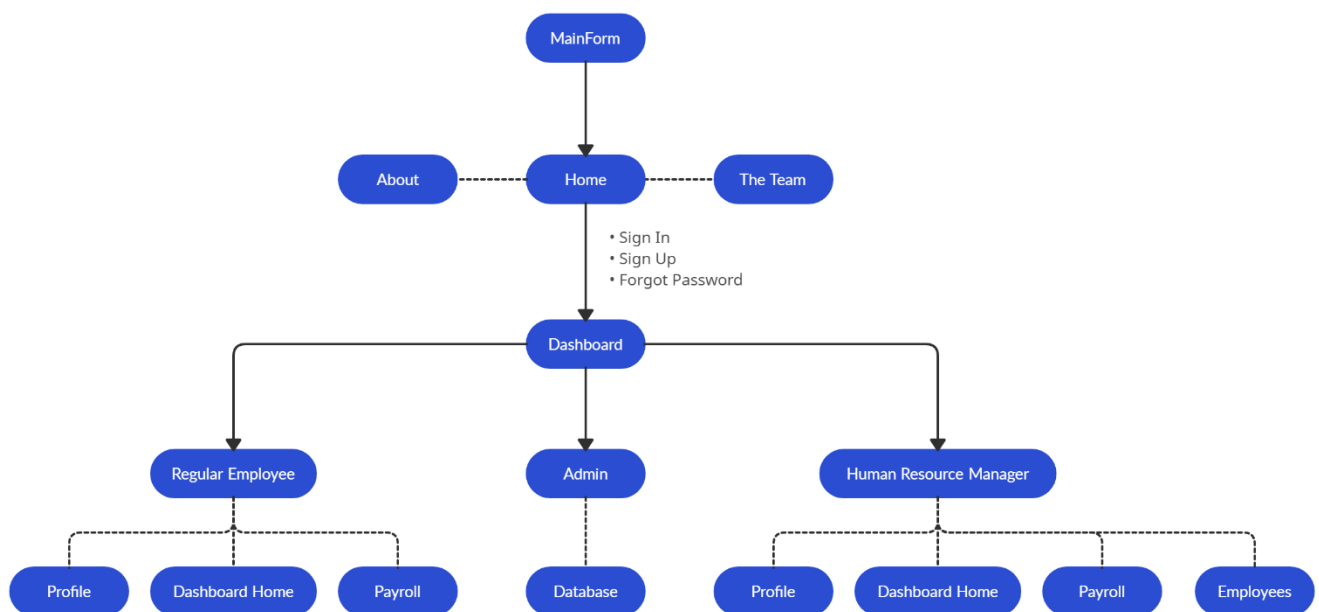
## G. Related Documents

- [TimyVillarmia/Dashboard: CS201 Dashboard is a modern-like design Windows Forms Application develop in C# and .NET Framework \(github.com\)](#)
- [Introduction \(mimekit.net\)](#)
- [Command Line Shell For SQLite](#)
- [SQLite Database: How to Create, Open, Backup & Drop Files \(guru99.com\)](#)
- [Overview - Microsoft.Data.Sqlite | Microsoft Learn](#)
- [Devart.Common Namespace \(Devart.Data\)](#)
- [c# - When to use ExecuteScalar, ExecuteReader, and ExecuteNonQuery? - Stack Overflow](#)
- [C# SQLite Parameterized Select Using LIKE - Stack Overflow](#)

- [C# SQLite insert, update, delete data | Source Code](#)
- [Connect C# Application to SQLite Database and interact with it](#)
- [SQLite Tutorial For Beginners - Make A Database In No Time](#)
- [Guna UI WinForms | Guna Framework - .NET UI/UX Controls and Components for Developers of Desktop, Reporting, Data Visualization](#)

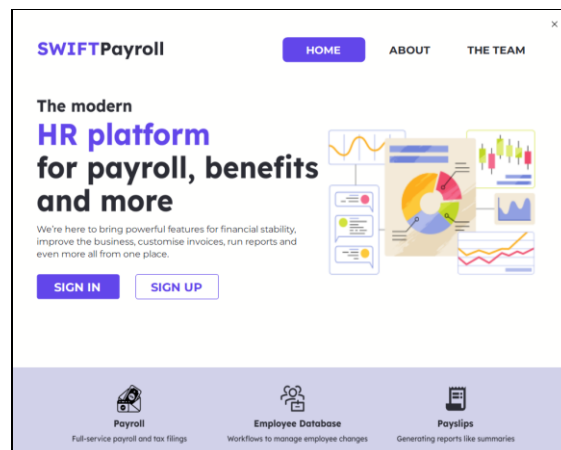
## II. Design

### A. Use-Case Diagram



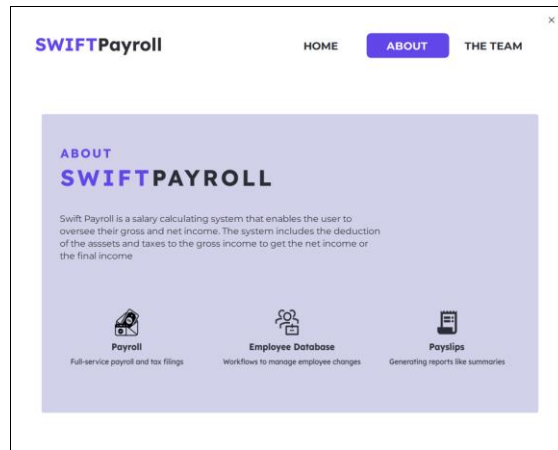
### B. User Interface

#### 1. Main Form Home

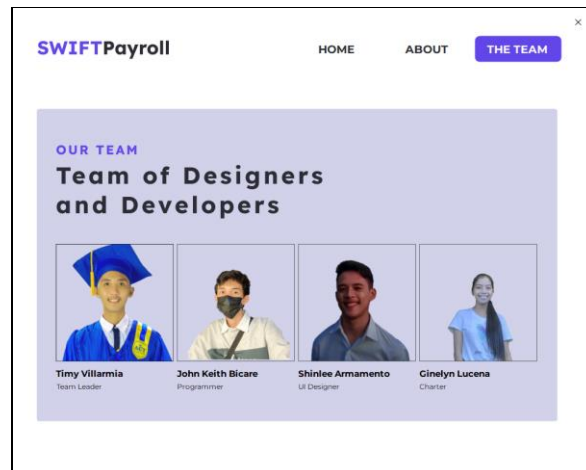




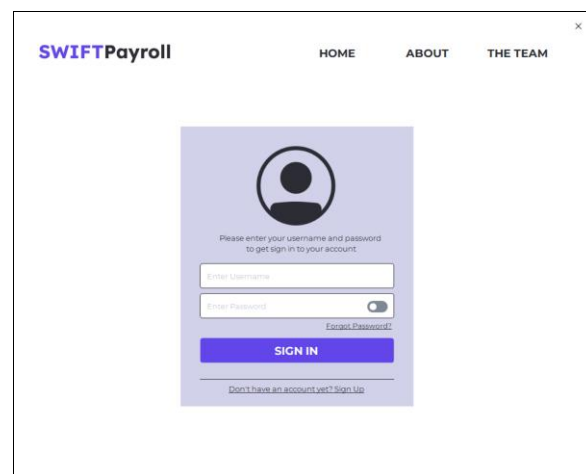
## 2. About Page



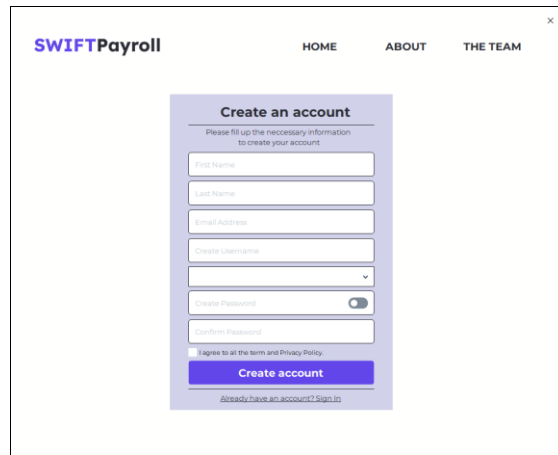
## 3. The Team Page



## 4. Sign In Page

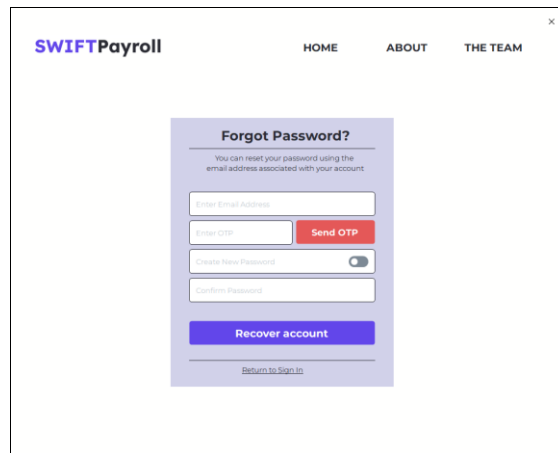


## 5. Sign Up Page



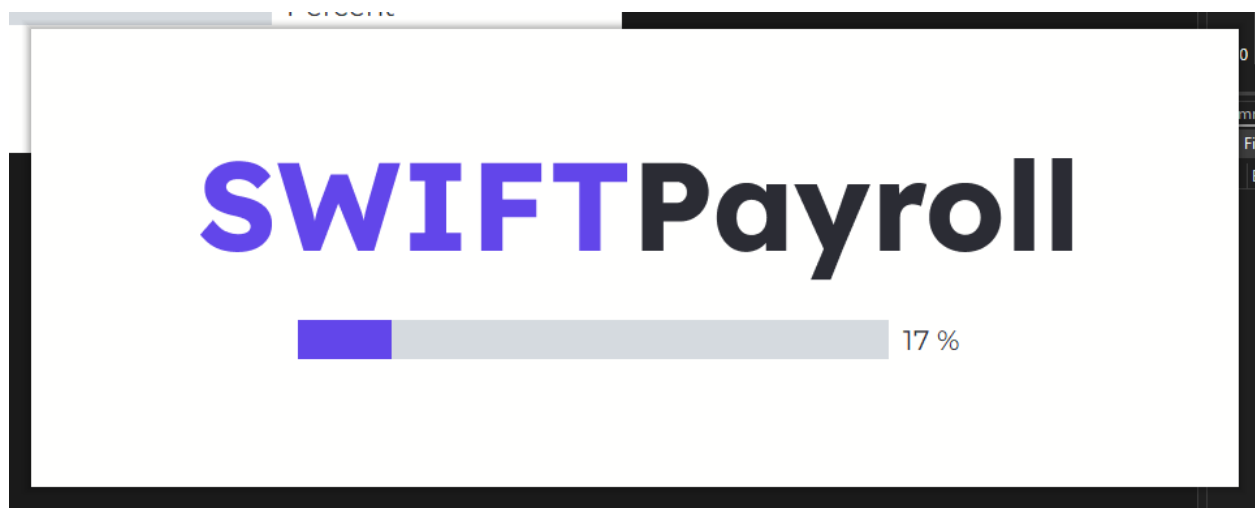
The Sign Up page for SWIFT Payroll features a clean, modern design. At the top, the SWIFT Payroll logo is on the left, and navigation links for HOME, ABOUT, and THE TEAM are on the right. The main content area is a light purple box titled "Create an account". Below the title, a subtitle reads "Please fill up the necessary information to create your account". The form includes input fields for First Name, Last Name, Email Address, and Create Username. There is a dropdown menu for selecting a role, a password field with a toggle for "Create Password", and a "Confirm Password" field. A checkbox for "I agree to all the term and Privacy Policy" is present. A prominent purple "Create account" button is at the bottom of the form. A link "Already have an account? Sign in" is located below the button.

## 6. Forgot Password Page



The Forgot Password page for SWIFT Payroll has a similar layout to the Sign Up page. It features the SWIFT Payroll logo and navigation links at the top. The main content area is a light purple box titled "Forgot Password?". Below the title, a subtitle reads "You can reset your password using the email address associated with your account". The form includes an input field for "Enter Email Address", a "Send OTP" button, a "Create New Password" field with a toggle, and a "Confirm Password" field. A prominent purple "Recover account" button is at the bottom of the form. A link "Return to Sign in" is located below the button.

## 7. Loading Screen



## 8. Dashboard (Admin)

ADMIN

General

Database

Sign Out

Employee ID	First Name	Last Name	Username	Password	Email Address	DOB	Gender
2022-9999	Timy	Villarmia	TimyHR	TimyHR	TimyHR	EM	Male
2022-12016	Timy	Villarmia	Villarmia	Villarmia	Villarmia	EM	Male

Timy Villarmia

Employee ID: 2022-9999

Email: TimyHR

Contact Number

Current Position: Human Resources Manager

Type: Full-Time

Department

Username:

Password:

Save

## 9. Dashboard (Human Resources Manager)

TimyHR

General

Dashboard

Profile

Payroll

Employees

Sign Out

**Dashboard**

Hello Timy Villarmia

Welcome to your overview of your account

**Performance**

2022

Worked Days: 257

Overtime: 20

Absences: 5

Late: 1

**Notifications**

**Board Meeting**

Dec 07 at 5:00 PM

You have been invited to attend a meeting of the Board Directors.

Join Meeting

**To Do List**

Timy Villarmia

Employee ID: 2022-12016

Current Position: Administrative

Type: Full-Time

Pay Date: Friday, 25 November 2022

Overtime: 0

Over Time: 0

Worked Days: 0

Absences: 0

**EARNINGS**

STANDARD PAY	RATE	HOURS	CURRENT
Standard Pay	PHP 8 hrs	0	0
Overtime	0	0	0
Gross Income:			PHP 0.00

**DEDUCTIONS**

SSS	PAG-IBIG	PHILHEALTH	Absences	TAX	CURRENT
0	0	0	0	10%	0
Net Income:					PHP 0.00

Generate Payslip

TimyHR

General

Dashboard

Profile

Payroll

Employees

Sign Out

**Timy Villarmia**

Employee ID: 2022-9999

Current Position: Human Resources Manager

Type: Part-Time

Department

Email: TimyHR

Contact Number

First Name: Timy

Last Name: Villarmia

Username: TimyHR

Password: TimyHR

Sex: Male

Contact: TimyHR

Email: TimyHR

Address: TimyHR

Edit

TimyHR

General

Dashboard

Profile

Payroll

Employees

Sign Out

**Payroll**

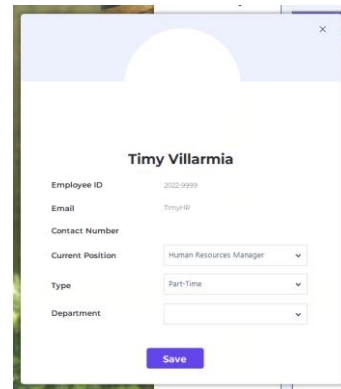
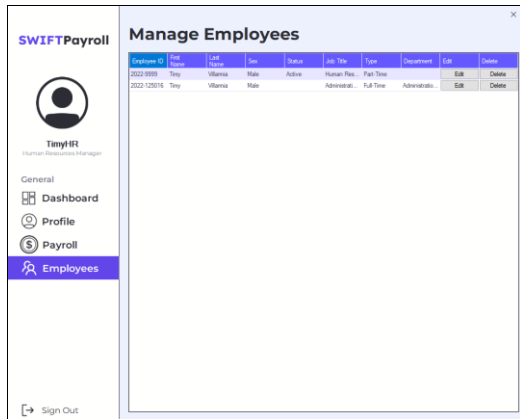
Employee ID	First Name	Last Name	Sex	Status	Job Title	Type	Department
2022-9999	Timy	Villarmia	Male	Active	Human Resources	Full-Time	Administration
2022-12016	Timy	Villarmia	Male	Active	Human Resources	Full-Time	Administration

Employee ID:

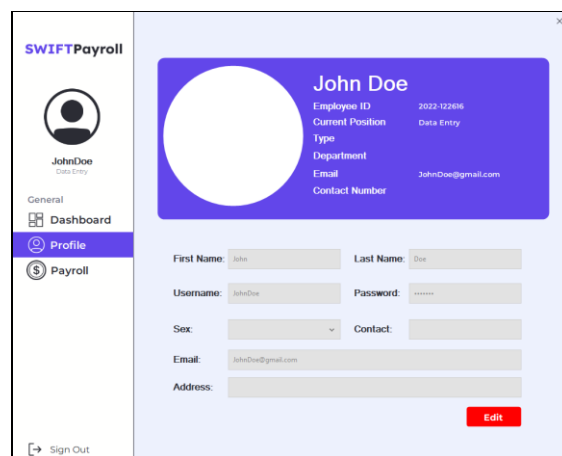
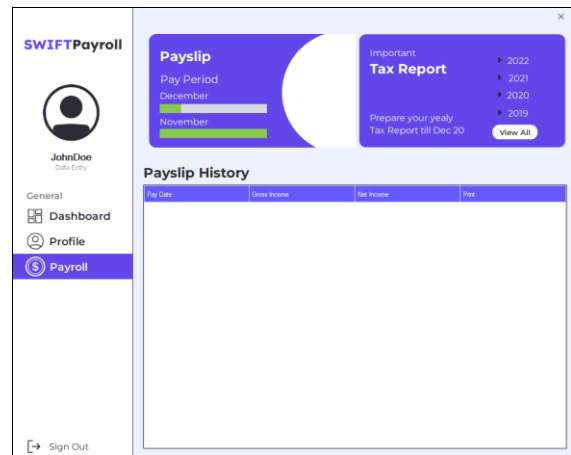
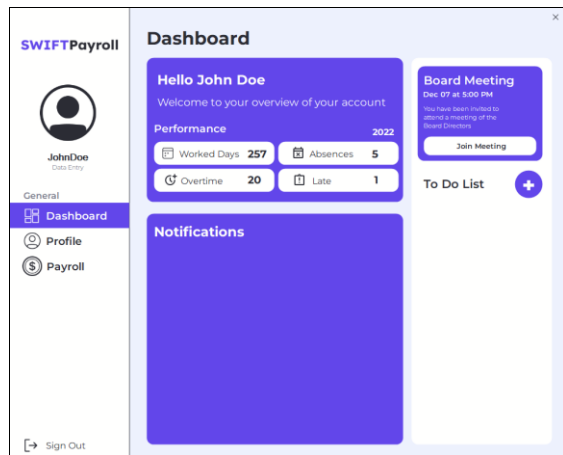
VIEW

**Recent Activity**

Employee ID	First Name	Last Name	Job Title	Net Income
2022-9999	Timy	Villarmia	Human Resources	PHP 0.00
2022-12016	Timy	Villarmia	Human Resources	PHP 0.00



## 10. Dashboard (Regular Employee)



## 11. Pay slip PDF

**SWIFTPayroll**

Employee Information

Name: Timy Villamia

Employee ID: 2023-125016

Current Position: Administrative Assistant

Type: Full-Time

Worked Days: 30

Pay Date: 25-11-2022

Department: Administration department

Email: Villamiaque

Contact Number: wqe

Absences: 12

## III. Software Specification

### ABOUT PROJECT

Project Name:  
Project Platform:  
Programming Language Used:  
IDE Tool (Recommended):  
Project Type:  
Database:

### PROJECT DETAILS

SwiftPayroll  
Windows Forms App (.NET Framework 4.8)  
C#  
Visual Studio 2022  
Desktop Applications  
SQLite

**Username: ADMIN Password: ADMIN**

### A. NuGet Packages and APIs

#### 1. Guna.UI2.WinForms

Guna UI is the suite for creating groundbreaking desktop app UI. It is for developers targeting the .NET Windows Forms platform. Guna UI guarantees faster development and improved productivity.

### Requirements

#### Minimum System and User Requirements

Guna UI currently work with C# and VB.NET languages, specifically and specially designed for Windows Forms applications.

To effectively get the most out of the tools, your system needs to meet the following minimum requirements:

- Runs on Windows.
- Microsoft .NET Framework 4.0 or higher.
- Visual Studio 2012 or later (recommended).

## 2. MailKit & MimeKit

MailKit is an open-source cross-platform email framework for the .NET platform. It can be used to build applications that send, receive, and process email messages. It is built on top of the MimeKit library, which provides the low-level logic for parsing and generating email messages. MailKit is designed to be easy to use and extensible, and it provides many advanced features such as support for the IMAP, STMP, and POP3 protocols.

## 3. ITextSharp

ITextSharp is a .NET library that allows developers to create, read, and manipulate PDF documents in C# and VB.NET. It is an open-source library, and it provides a simple and intuitive API for working with PDF files. ITextSharp is widely used in the development of applications that need to generate or manipulate PDF documents, such as document management systems, reporting tools, and e-commerce applications.

## 4. System.Data.SQLite

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world.

System.Data.SQLite is a data provider for the SQLite database engine. It allows developers to integrate SQLite with the .NET Framework, providing a simple and intuitive API for working with SQLite databases. System.Data.SQLite is an open-source library, and it is widely used in the development of applications that need to store and manipulate data in SQLite databases, such as desktop and mobile applications.

## IV. Software Implementation

### A. EmployeeInfo Class

```
public class EmployeeInfo
{
    //Employee Infomation
    public string EmployeeID { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Fullname
    {
        get { return FirstName + " " + LastName; }
        set { FirstName = value; }
    }
    public string Sex { get; set; }
    public string Status { get; set; }
    public string Username { get; set; }
    public string Password { get; set; }
    public string Email { get; set; }
    public string ContactNumber { get; set; }
    public string Address { get; set; }
    public string Title { get; set; }
    public string Type { get; set; }
    public string Department { get; set; }

    // Employee's Payslip Information
    public string PayDate { get; set; }
    public string WorkedDay { get; set; }
    public string AbsencesDay { get; set; }
    public string StandardPay { get; set; }
    public string OvertimeHours { get; set; }
    public string OvertimePay { get; set; }
    public string GrossIncome { get; set; }
    public string SSS { get; set; }
    public string PagIbig { get; set; }
    public string PhilHealth { get; set; }
    public string AbsencesDeduction { get; set; }
    public string Tax { get; set; }
    public string NetIncome { get; set; }

    public EmployeeInfo()
    {
    }

    //method for generating employee ID
    public string GenerateEmployeeID()
    {
        string month, year;
        string min, sec;
        month = DateTime.Now.Month.ToString();
        year = DateTime.Now.Year.ToString();

        min = DateTime.Now.Minute.ToString();
        sec = DateTime.Now.Second.ToString();

        return EmployeeID = year + "-" + month + min + sec;
    }
}
```

```

//method for getting all the personal information of an employee
public void GetInformation(string username)
{
    using (var connection = new SQLiteConnection(@"Data Source=Database\Accounts.db"))
    {
        connection.Open();
        string query = "SELECT * FROM Accounts WHERE username=@Username;";

        using (var command = new SQLiteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Username", username);
            SQLiteDataReader data = command.ExecuteReader();
            data.Read();

            EmployeeID = $"{data["employeeID"]}";
            FirstName = $"{data["firstname"]}";
            LastName = $"{data["lastname"]}";
            Sex = $"{data["sex"]}";
            Status = $"{data["status"]}";
            Username = $"{data["username"]}";
            Password = $"{data["password"]}";
            Email = $"{data["email"]}";
            ContactNumber = $"{data["contactnumber"]}";
            Address = $"{data["address"]}";
            Title = $"{data["title"]}";
            Type = $"{data["type"]}";
            Department = $"{data["department"]}";

            data.Close();
        }
    }
}

//method for logging in
public int Login(string username, string password)
{
    using (var connection = new SQLiteConnection(@"Data Source=Database\Accounts.db"))
    {
        connection.Open();
        string query = "SELECT count(*) FROM Accounts WHERE username = @Username AND password = @Password";

        using (var command = new SQLiteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Username", username);
            command.Parameters.AddWithValue("@Password", password);
            int count = Convert.ToInt32(command.ExecuteScalar());
            //if count = 1 then the account exist; else account doesn't exist
            return count;
        }
    }
}

```



```

//method for registering an account
public void Register(string firstname, string lastname, string username, string password,
string email, string title)
{
    using (var connection = new SQLiteConnection(@"Data Source=Database\Accounts.db"))
    {
        connection.Open();

        string query = "INSERT INTO Accounts(employeeID,firstname,lastname,username,password,email,title) VALUES(@employeeid,@first,@last,@username,@password,@email,@title);";

        using (var command = new SQLiteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@employeeid", GenerateEmployeeID());
            command.Parameters.AddWithValue("@first", firstname);
            command.Parameters.AddWithValue("@last", lastname);
            command.Parameters.AddWithValue("@username", username);
            command.Parameters.AddWithValue("@password", password);
            command.Parameters.AddWithValue("@email", email);
            command.Parameters.AddWithValue("@title", title);
            command.ExecuteNonQuery();
            MessageBox.Show("Account successfully created, Please Sign In");
        }
    }
}

//method for recovering an account
public void RecoverAccount(string email,string password)
{
    using (var connection = new SQLiteConnection(@"Data Source=Database\Accounts.db"))
    {
        connection.Open();
        string query = "UPDATE Accounts SET password=@Password WHERE email = @Email;";

        using (var command = new SQLiteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Email", email);
            command.Parameters.AddWithValue("@Password", password);
            command.ExecuteNonQuery();
            MessageBox.Show("Account successfully reset, Please Sign In");
        }
    }
}

```

```

//method for updating employee's personal information
public void UpdateProfile(string firstname, string lastname, string password, string sex,
string contact, string email, string address)
{
    using (var connection = new SQLiteConnection(@"Data Source=Database\Accounts.db"))
    {
        connection.Open();
        string query = "UPDATE Accounts SET
sex=@sex,firstname=@first,lastname=@last,password=@password,email=@email,contactnumber=@contact,a
ddress=@address WHERE username=@currentuser;";

        using (var command = new SQLiteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@sex", sex);
            command.Parameters.AddWithValue("@first", firstname);
            command.Parameters.AddWithValue("@last", lastname);
            command.Parameters.AddWithValue("@currentuser", Username);
            command.Parameters.AddWithValue("@password", password);
            command.Parameters.AddWithValue("@email", email);
            command.Parameters.AddWithValue("@contact", contact);
            command.Parameters.AddWithValue("@address", address);
            command.ExecuteNonQuery();
        }
    }
}

//method for getting payslip of an employee
public void GetPaySlip(string paydate)
{
    using (var connection = new SQLiteConnection(@"Data Source=Database\PaySlip.db"))
    {
        connection.Open();
        string query = "SELECT * FROM EmployeePaySlip WHERE EmployeeID=@employeeID AND
PayDate=@paydate;";
        using (var command = new SQLiteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@employeeID", EmployeeID);
            command.Parameters.AddWithValue("@paydate", paydate);

            using (SQLiteDataReader data = command.ExecuteReader())
            {
                data.Read();

                WorkedDay = $"{data["WorkedDay"]}";
                AbsencesDay = $"{data["AbsencesDay"]}";
                StandardPay = $"{data["StandardPay"]}";
                OvertimeHours = $"{data["OvertimeHours"]}";
                OvertimePay = $"{data["OvertimePay"]}";
                GrossIncome = $"{data["GrossIncome"]}";
                SSS = $"{data["SSS"]}";
                PagIbig = $"{data["PagIbig"]}";
                PhilHealth = $"{data["PhilHealth"]}";
                AbsencesDeduction = $"{data["AbsencesDeduction"]}";
                Tax = $"{data["Tax"]}";
                NetIncome = $"{data["NetIncome"]}";
            }
        }
    }
}

```

## B. Sign In Page

```
private void SignInBtn_Click(object sender, EventArgs e)
{
    //SQLiteConnection connection = new SQLiteConnection(@"Data Source=Database\Accounts.db")
    // check login attempts
    // if attempt != 0 - authenticate
    // if attempt == 0 - login lock for 60 seconds
    if (attempt != 0)
    {
        //check if textboxes are not empty
        if (UsernameTxt.Text.Trim() == string.Empty && PasswordTxt.Text.Trim() == string.Empty)
        {
            MessageBox.Show("Make sure you correctly fill up the form");
        }

        // admin authentication
        if (UsernameTxt.Text == "ADMIN" && PasswordTxt.Text == "ADMIN")
        {
            currentuser = UsernameTxt.Text;
            //notify
            MessageBox.Show("Login Successfully");
            // hide the MainForm
            ParentForm.Hide();
            // displaying sencond form "loading screen form"
            LoadingScreenForm loading = new LoadingScreenForm();
            loading.ShowDialog();
            //Closing
            ParentForm.Close();
            //close connection

        }
        else // non-admin authentication
        {
            try
            {
                EmployeeInfo employee = new EmployeeInfo();

                //call the login method from EmployeeInfo class
                // Login(argument 1, argument 2)
                if (Employee.Login(UsernameTxt.Text.Trim(), PasswordTxt.Text.Trim()) == 1)
                {
                    currentuser = UsernameTxt.Text;

                    //notify
                    MessageBox.Show("Login Successfully");
                    // hide the MainForm
                    ParentForm.Hide();

                    // displaying sencond form "loading screen form"
                    LoadingScreenForm loading = new LoadingScreenForm();
                    loading.ShowDialog();
                    //Closing
                    ParentForm.Close();

                    //// Clear all entries
                    UsernameTxt.Text = "";
                    PasswordTxt.Text = "";

                }
                else
                {
                    //notify
                    attempt -= 1;
                    MessageBox.Show($"Wrong username and password combination. {attempt} Attempts left");
                }

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
    else
    {
        //Start timer if attempts = 0
        timer1.Start();
    }
}
```

```

private void timer1_Tick(object sender, EventArgs e)
{
    seconds -= 1; // decrement by 1
    if (seconds != 0)
    {
        label1.Visible = true; // show timer text
        label2.Visible = false; // hide default text
        SignInBtn.Enabled = false; //unclickable button
        loginPictureBox.Image = SwiftPayroll.Properties.Resources.locked; // change photo to lock
        label1.Text = $"Too many failed login attempts\r\n Please try again after {seconds}"; // timer
        label1.ForeColor = Color.Red; // change to red forecolor
    }
    else
    {
        label1.Visible = false; //hide timer text
        label2.Visible = true; // show default text
        SignInBtn.Enabled = true; //clickable button
        loginPictureBox.Image = SwiftPayroll.Properties.Resources.profile; // change to default photo
        timer1.Stop(); //stop timer if it reaches to 0
        seconds = 60; // reset to 60 seconds
        attempt = 3; // reset to 3 attempts
    }
}

```

## C. Sign Up Page

```

private void CreateAccountBtn_Click(object sender, EventArgs e)
{
    //check if data entries exist
    if (FirstNameTxt.Text == string.Empty && LastNameTxt.Text == string.Empty && EmailTxt.Text == string.Empty && Username-
Txt.Text == string.Empty && PasswordTxt.Text == string.Empty && ConfirmPasswordTxt.Text == string.Empty)
    {
        //notify
        MessageBox.Show("Make sure you correctly fill up the form");
    }
    if (PasswordTxt.Text != ConfirmPasswordTxt.Text)
    {
        MessageBox.Show("Passwords are not identical");
    }
    if (TermsCheck.Checked == false)
    {
        MessageBox.Show("Don't forget to check the Terms and Privacy Policy");
    }
    else
    {
        try
        {
            EmployeeInfo employee = new EmployeeInfo();

            //call the register method from the EmployeeInfo class
            employee.Register(FirstNameTxt.Text.Trim(), LastNameTxt.Text.Trim(), UsernameTxt.Text.Trim(), Pass-
wordTxt.Text.Trim(), EmailTxt.Text.Trim(), TitleComboBox.SelectedItem.ToString());

            // Clear all entries
            FirstNameTxt.Text = "";
            LastNameTxt.Text = "";
            EmailTxt.Text = "";
            UsernameTxt.Text = "";
            PasswordTxt.Text = "";
            ConfirmPasswordTxt.Text = "";
            TitleComboBox.SelectedIndex = -1;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            //notify
            MessageBox.Show("Make sure the Username is unique and Email is not currently registered");
        }
    }
}

```

## D. Forgot Password Page

```
private void RecoverBtn_Click(object sender, EventArgs e)
{
    //Guard Clause Technique
    if (EmailTxt.Text == string.Empty && PasswordTxt.Text == string.Empty && ConfirmPasswordTxt.Text == string.Empty &&
    OTPTxt.Text == string.Empty)
    {
        MessageBox.Show("Make sure you correctly fill up the form");
    }
    if (OTPTxt.Text != OTP)
    {
        MessageBox.Show("Invalid OTP");
    }
    if (PasswordTxt.Text != ConfirmPasswordTxt.Text)
    {
        MessageBox.Show("Passwords are not identical");
    }
    else
    {
        try
        {
            EmployeeInfo employee = new EmployeeInfo();
            //call the RecoverAccount method from EmployeeInfo class
            employee.RecoverAccount(EmailTxt.Text.Trim(), PasswordTxt.Text.Trim());

            // Clear all entries
            EmailTxt.Text = "";
            OTPTxt.Text = "";
            PasswordTxt.Text = "";
            ConfirmPasswordTxt.Text = "";
        }
        catch (Exception)
        {
            //notify
            MessageBox.Show("Unable to reset your password, please try again");
        }
    }
}
```

## E. One-Time-Password Event

```
//method for generating OTP
private string GenerateOTP()
{
    //OTP generator
    string otp_char = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
    OTP = "";
    Random rnd = new Random();

    for (int i = 0; i < 6; i++)
    {
        var random_char = otp_char[rnd.Next(1, otp_char.Length)];
        OTP += random_char;
    }
    return OTP;
}

//method for sending OTP
private void OTPBtn_Click(object sender, EventArgs e)
{
    using (var connection = new SQLiteConnection(@"Data Source=Database\Accounts.db"))
    {
        connection.Open();
        string query = "SELECT count(*) FROM Accounts WHERE email = @Email";
        using (var command = new SQLiteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Email", EmailTxt.Text);
            int count = Convert.ToInt32(command.ExecuteScalar());

            if (count == 1)
            {
                string msg = GenerateOTP();
                string senderEmail, senderPass, receiverEmail;
                receiverEmail = EmailTxt.Text;
                senderEmail = "Sender's Gmail Email Address"; //Change this to your Sender's Gmail Email Address
                senderPass = "Sender's Gmail App Password"; //Gmail's App Password Change this to your Sender's Gmail App
                Password

                MimeMessage message = new MimeMessage(); // Creating object for Message
                message.From.Add(new MailboxAddress("SwiftPayroll - OTP", senderEmail)); //Sender's information
                message.To.Add(MailboxAddress.Parse(receiverEmail)); //Receiver's Information

                message.Subject = "One-Time-Password"; //Email's Subject

                //Email's Body
                message.Body = new TextPart("plain") //Plain text
                {
                    Text = msg //MSG = OTP
                };

                SmtpClient client = new SmtpClient(); // allows sending of e-mail notifications using a SMTP server

                try
                {
                    client.Connect("smtp.gmail.com", 465, true); //Gmail's smtp server, PORT: 465
                    client.Authenticate(senderEmail, senderPass); //Login sender's email and password
                    client.Send(message); //
                    MessageBox.Show("Kindly check your email and don't forget to check your SPAM folders");
                }
                catch (Exception)
                {
                    MessageBox.Show("Unable to generate OTP, please try again");
                    return;
                }
                finally
                {
                    client.Disconnect(true); // always Disconnect the service.
                    client.Dispose(); //Releases all resource used by the MailService object.
                }
            }
            else
            {
                MessageBox.Show("Email is not registered");
            }
        }
    }
}
```

## F. Pay slip calculation and generate PDF

```
private void CalculatePaySlip()
{
    //calculations
    double GrossIncome = StandardPay + OvertimeIncome;
    double TaxTotal = GrossIncome * .1;
    double TotalAbsentDeduction = 200 * NumofAbsent;
    double TotalDeduction = SSS + PAGIBIG + PHILHEALTH + TotalAbsentDeduction + TaxTotal;
    double NetIncome = GrossIncome - TotalDeduction;

    //displaying
    GrossIncomeLbl.Text = String.Format("P{0:n}", GrossIncome);
    TaxLbl.Text = String.Format("P{0:n}", TaxTotal);
    DeductionsLBL.Text = String.Format("P{0:n}", TotalDeduction);
    NetIncomeLbl.Text = String.Format("P{0:n}", NetIncome);
}

//method for creating PDF
public void GeneratePDFReports()
{
    string paydate = Selected_PayslipDate;
    string Standard_HourlyRate = "-";
    string Standard_Hours = "-";
    string OT_Rate = "P45";
    string Absences_Rate = "P200";

    employee = new EmployeeInfo();

    try
    {
        // For getting data from Accounts.db
        employee.GetInformation(currentuser.CurrentUser);
        // For getting data from PaySlip.db
        employee.GetPaySlip(paydate);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }

    if (employee.Type == "Full-Time")
    {
        Standard_HourlyRate = "P150";
        Standard_Hours = "8 hours";
    }
    else if (employee.Type == "Part-Time")
    {
        Standard_HourlyRate = "P80";
        Standard_Hours = "4 hours";
    }

    // create an instance of SaveFileDialog class
    SaveFileDialog saveFileDialog1 = new SaveFileDialog();
    //Title property is used to set or get the title of the open file dialog.
    saveFileDialog1.Title = "Save PDF";
    // Filter property represents the filter on an open file dialog that is used to filter the type of
    files to be loaded during the browse option in an open file dialog.
    saveFileDialog1.Filter = "PDF document (*.pdf)|*.pdf";
    //FileName property represents the file name selected in the open file dialog.
    saveFileDialog1.FileName = $"{paydate}_{employee.EmployeeID}_Payslip";

    //ShowDialog method displays the SaveFileDialog.
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        // Saves the PDF via a FileStream
        using (FileStream stream = new FileStream(saveFileDialog1.FileName + ".pdf", FileMode.Create))
        {
            // Create an instance of the document class which represents the PDF document itself with a
            page size of letter.
            Document document = new Document(PageSize.LETTER);
            // Create an instance to the PDF file by creating an instance of the PDF
            // using the document and the filestream in the constructor.
            PdfWriter.GetInstance(document, stream);
            //open document to be able to write contents
            document.Open();

            //directory for the logo
            string relativePathToLogo = Path.Combine(Directory.GetCurrentDirectory(), "Resources",
"SWIFTPayroll.png");
            //logo
            var logo = iTextSharp.text.Image.GetInstance(relativePathToLogo);
```

```

/*
    Table for the Employee's Information
*/

//creating a new table
PdfPTable InfoTable = new PdfPTable(2);

//default cell properties
InfoTable.DefaultCell.Border = iTextSharp.text.Rectangle.NO_BORDER;
InfoTable.DefaultCell.PaddingTop = 5;
InfoTable.DefaultCell.PaddingBottom = 5;
InfoTable.WidthPercentage = 100;
InfoTable.SpacingBefore = 20f;

//customized cell properties
PdfPCell cell = new PdfPCell(new Phrase("Employee Information"));
cell.PaddingBottom = 5;
cell.Colspan = 2;
//cell.HorizontalAlignment 0=Left, 1=Centre, 2=Right
cell.HorizontalAlignment = 1;
cell.Border = iTextSharp.text.Rectangle.NO_BORDER;
cell.BackgroundColor = new BaseColor(237, 241, 253);

//adding the individual cell to the table

//add customized cell to the table
InfoTable.AddCell(cell);
//default cells
//first row
InfoTable.AddCell($"Name: {employee.Fullname}");
InfoTable.AddCell($"Pay Date: {Selected_PayslipDate}");
//second row
InfoTable.AddCell($"Employee ID: {employee.EmployeeID}");
InfoTable.AddCell($"Department: {employee.Department}");
//third row
InfoTable.AddCell($"Current Position: {employee.Title}");
InfoTable.AddCell($"Email: {employee.Email}");
//fourth row
InfoTable.AddCell($"Type: {employee.Type}");
InfoTable.AddCell($"Contact Number: {employee.ContactNumber}");
//fifth row
InfoTable.AddCell($"Worked Days: {employee.WorkedDay}");
InfoTable.AddCell($"Absences: {employee.AbsencesDay}");

/*
    Table for the Earnings
*/

PdfPTable EarningTable = new PdfPTable(4);
//default cell properties
EarningTable.DefaultCell.PaddingTop = 5;
EarningTable.DefaultCell.PaddingBottom = 5;
EarningTable.WidthPercentage = 100;
EarningTable.SpacingBefore = 20f;
EarningTable.DefaultCell.HorizontalAlignment = 1;

//Column Headers
PdfPCell EarningHeader = new PdfPCell(new Phrase("EARNINGS"));
PdfPCell Rate_Header = new PdfPCell(new Phrase("RATE"));
PdfPCell Hours_Header = new PdfPCell(new Phrase("HOURS"));
PdfPCell Current_Header = new PdfPCell(new Phrase("CURRENT"));

//Column Headers

//Earnings Header Properties
EarningHeader.PaddingBottom = 5;
//cell.HorizontalAlignment 0=Left, 1=Centre, 2=Right
EarningHeader.HorizontalAlignment = 1;
EarningHeader.BackgroundColor = new BaseColor(237, 241, 253);

//Rate Header Properties
Rate_Header.PaddingBottom = 5;
//cell.HorizontalAlignment 0=Left, 1=Centre, 2=Right
Rate_Header.HorizontalAlignment = 1;
Rate_Header.BackgroundColor = new BaseColor(237, 241, 253);

//Hours Header Properties
Hours_Header.PaddingBottom = 5;
//cell.HorizontalAlignment 0=Left, 1=Centre, 2=Right
Hours_Header.HorizontalAlignment = 1;
Hours_Header.BackgroundColor = new BaseColor(237, 241, 253);

```



```

//Current Header Properties
Current_Header.PaddingBottom = 5;
//cell.HorizontalAlignment 0=Left, 1=Centre, 2=Right
Current_Header.HorizontalAlignment = 1;
Current_Header.BackgroundColor = new BaseColor(237, 241, 253);

//adding the individual cell to the table
//Headers
EarningTable.AddCell(EarningHeader);
EarningTable.AddCell(Rate_Header);
EarningTable.AddCell(Hours_Header);
EarningTable.AddCell(Current_Header);
//first row
EarningTable.AddCell("Standard Pay");
EarningTable.AddCell($"{Standard_HourlyRate}");
EarningTable.AddCell($"{Standard_Hours}");
EarningTable.AddCell($"{employee.StandardPay}");
//second row
EarningTable.AddCell("Overtime");
EarningTable.AddCell($"{OT_Rate}");
EarningTable.AddCell($"{employee.OvertimeHours}");
EarningTable.AddCell($"{employee.OvertimePay}");
//third row
EarningTable.AddCell("");
EarningTable.AddCell("");
EarningTable.AddCell("Gross Income: ");
EarningTable.AddCell($"{employee.GrossIncome}");

/*
    Table for the Deductions
*/
PdfPTable DeductionTable = new PdfPTable(3);
//default cell properties
DeductionTable.DefaultCell.PaddingTop = 5;
DeductionTable.DefaultCell.PaddingBottom = 5;
DeductionTable.WidthPercentage = 100;
DeductionTable.SpacingBefore = 20f;
DeductionTable.DefaultCell.HorizontalAlignment = 1;

//Column Headers
PdfPCell Deduction_Header = new PdfPCell(new Phrase("DEDUCTIONS"));
PdfPCell Blank_Header = new PdfPCell(new Phrase("RATE"));
PdfPCell Current1_Header = new PdfPCell(new Phrase("CURRENT"));
//Final Row
PdfPCell NetRow = new PdfPCell(new Phrase("Net Income: "));
PdfPCell NetIncomeCell = new PdfPCell(new Phrase($"{employee.NetIncome}"));

//Column Headers

//Deduction Header Properties
Deduction_Header.PaddingBottom = 5;
//cell.HorizontalAlignment 0=Left, 1=Centre, 2=Right
Deduction_Header.HorizontalAlignment = 1;
Deduction_Header.BackgroundColor = new BaseColor(237, 241, 253);

//Blank Header Properties
Blank_Header.PaddingBottom = 5;
//cell.HorizontalAlignment 0=Left, 1=Centre, 2=Right
Blank_Header.HorizontalAlignment = 1;
Blank_Header.BackgroundColor = new BaseColor(237, 241, 253);

//Current1 Header Properties
Current1_Header.PaddingBottom = 5;
//cell.HorizontalAlignment 0=Left, 1=Centre, 2=Right
Current1_Header.HorizontalAlignment = 1;
Current1_Header.BackgroundColor = new BaseColor(237, 241, 253);

//Final Row Properties
NetRow.PaddingBottom = 5;
NetRow.Colspan = 2;
//cell.HorizontalAlignment 0=Left, 1=Centre, 2=Right
NetRow.HorizontalAlignment = 2;
NetRow.Border = iTextSharp.text.Rectangle.NO_BORDER;

//Net Income Cell Properties
NetIncomeCell.PaddingBottom = 5;
//cell.HorizontalAlignment 0=Left, 1=Centre, 2=Right
NetIncomeCell.HorizontalAlignment = 1;
NetIncomeCell.Border = iTextSharp.text.Rectangle.NO_BORDER;

//Headers
DeductionTable.AddCell(Deduction_Header);
DeductionTable.AddCell(Blank_Header);
DeductionTable.AddCell(Current1_Header);

```

```

//first row
DeductionTable.AddCell("SSS");
DeductionTable.AddCell($"{employee.SSS}");
DeductionTable.AddCell($"{employee.SSS}");
//second row
DeductionTable.AddCell("PAG-IBIG");
DeductionTable.AddCell($"{employee.PagIbig}");
DeductionTable.AddCell($"{employee.PagIbig}");
//third row
DeductionTable.AddCell("PHILHEALTH");
DeductionTable.AddCell($"{employee.PhilHealth}");
DeductionTable.AddCell($"{employee.PhilHealth}");
//fourth row
DeductionTable.AddCell("Absences");
DeductionTable.AddCell($"{Absences_Rate}");
DeductionTable.AddCell($"{employee.AbsencesDeduction}");
//fifth row
DeductionTable.AddCell("TAX");
DeductionTable.AddCell("10%");
DeductionTable.AddCell($"{employee.Tax}");
//final row
DeductionTable.AddCell(NetRow);
//Net Income Cell
DeductionTable.AddCell(NetIncomeCell);

//adding contents
document.Add(Logo);
document.Add(InfoTable);
document.Add(EarningTable);
document.Add(DeductionTable);

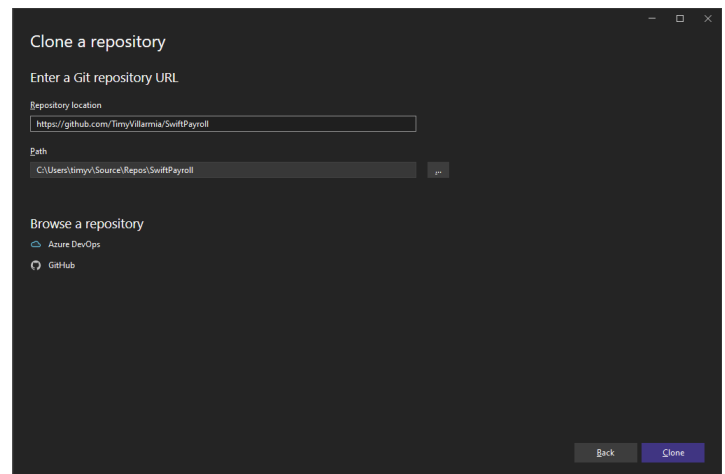
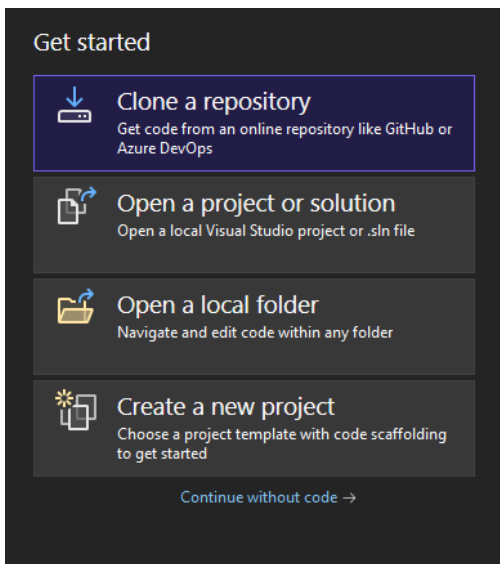
document.Close();
}
}
}

```

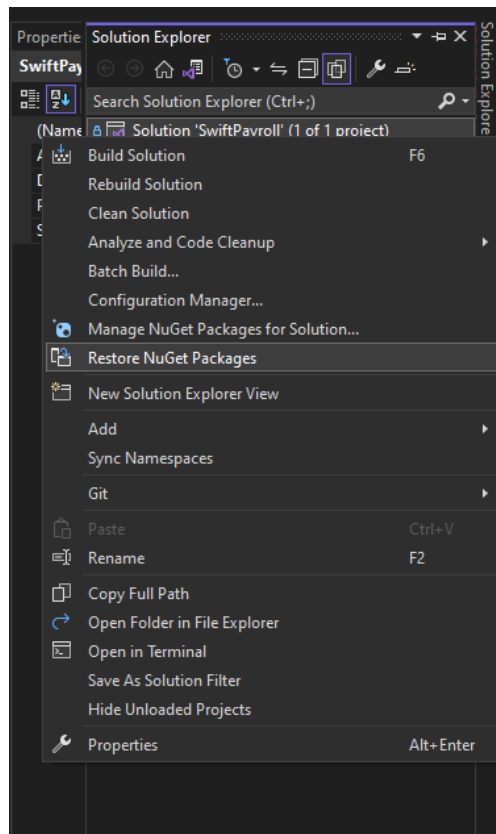
## G. How to run the Project

Step 1: clone the GitHub repository.

<https://github.com/TimyVillarmia/SwiftPayroll>



## Step 2: Restore NuGet Packages



## Step 3: Start project

