

GROUP

ASSIGNMENT

Module Code :

CT044-3-1-IOOP

Title : APU

Management System



Printing Services

Intake Code :

APU1FCS(IS)

Hand Out Date : 21 August 2023

Hand In Date : 17 November 2023

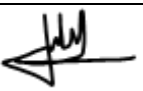


No	Name	TP Number	Signature
1	Tay Jun Long	TP074666	
2	Angelina Leanore	TP072929	
3	Terence Lim Dao Liang	TP073243	
4	Eraliev Suimonkul	TP068888	

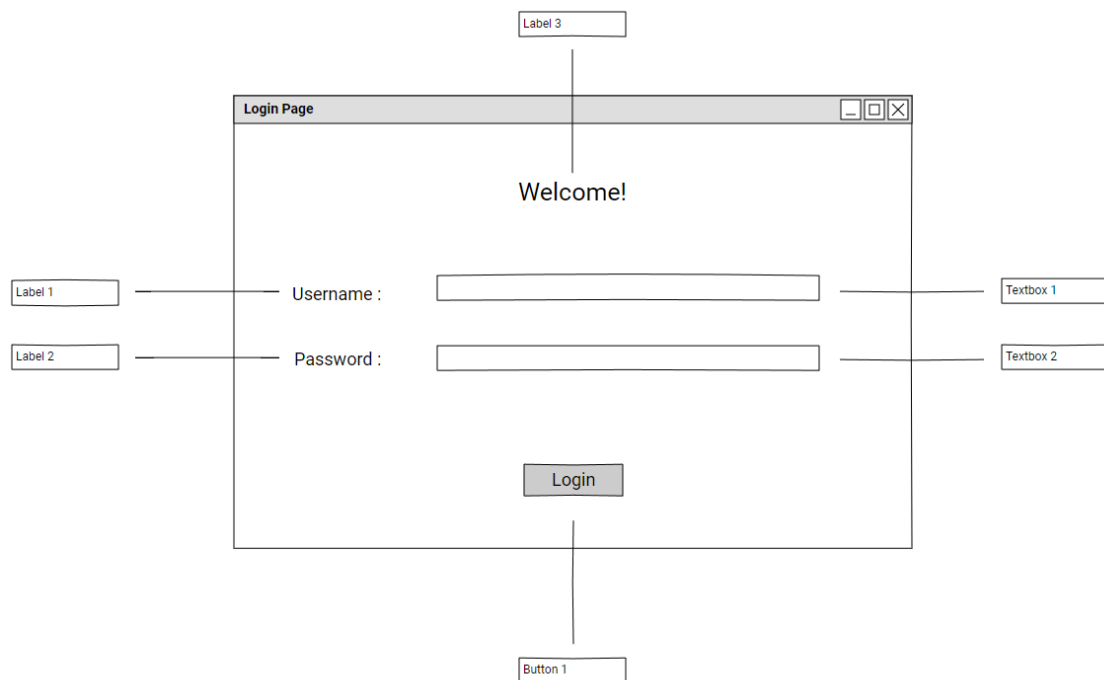
Table of Content

1. Introduction.....	3
2. Storyboard.....	4
3. Use-case Diagram.....	11
4. Class Diagram	12
4. Explanation of The implemented codes	13
Login page	13
Admin Main Page	17
Admin Register Form	18
Admin View Request Report.....	21
Admin Monthly Income Report.....	22
Customer Main Menu.....	24
Customer Pending.....	26
Customer New Request	28
Manager Main Menu	32
Receive Payment (Also Customer's Pay button).....	34
Manager Assign Worker.....	38
Worker Main Menu	40
Worker Status Update	42
5. Test Plans and Cases	44
6. Conclusion	47
7. References.....	48
8. Workload Matrix.....	49

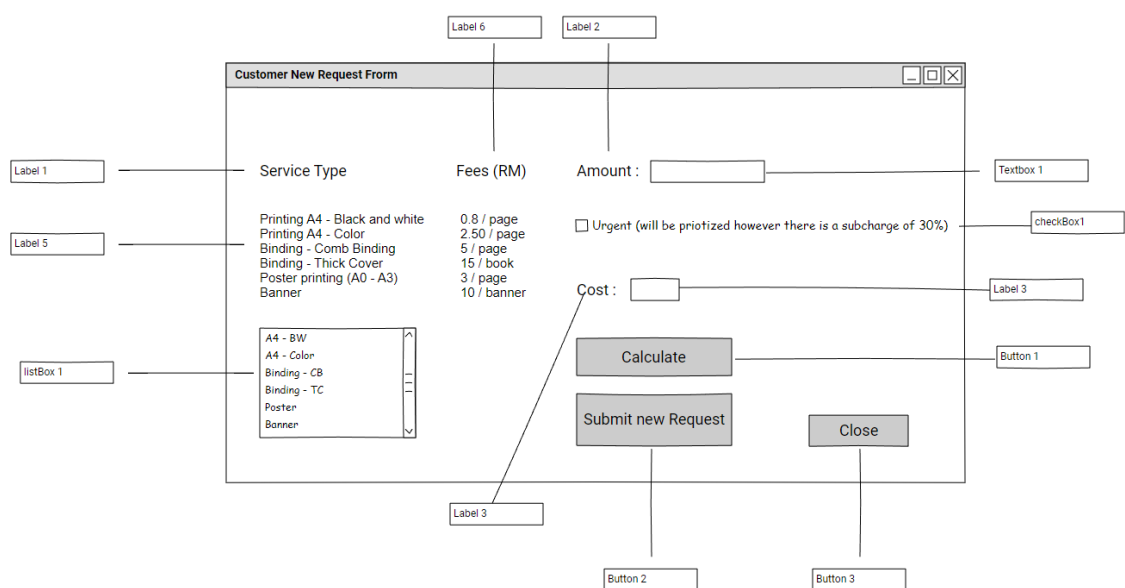
1. Introduction

This project was coded by C# and SQL to create a dynamic printing & binding services system for Students. All data of the User accounts, Service type and Request detail are store in a SQL server database. Object-oriented programming such as methods, classes and objects is used in the project to enhance the efficiency and optimization of the system. Connecting SQL database to C# is made possible with SQL Command in C#, which allows commands to be submitted to database via a query. The SQL connection object specifies the SQL command. There are two ways used: ExecuteNonQuery for insert, update, and delete instructions, and ExecuteReader method for query results (Thompson, 2023). It is the most effective approach for the various instructions. Data retrieve from the SQL database is either used directly as data source or put in a data table. A collection of columns and rows are provided by the DataTable class in C# ADO.NET, which is a database table representation that allows data to be stored in a grid format (Chand, 2023). Utilizing data table, the information display on the data grid view can be controlled.

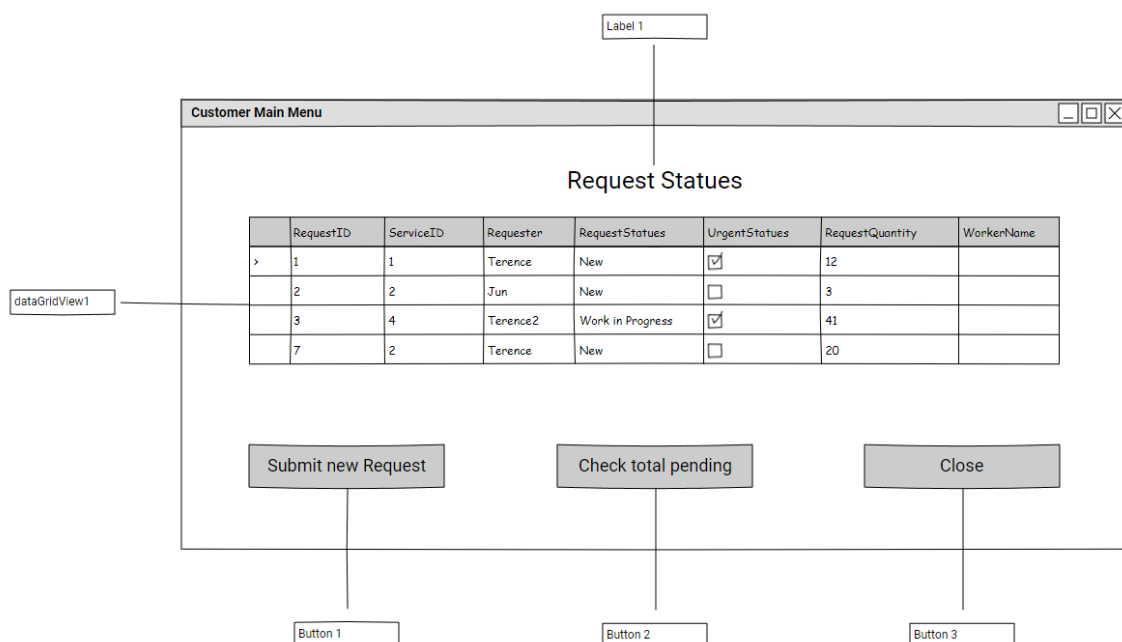
2. Storyboard



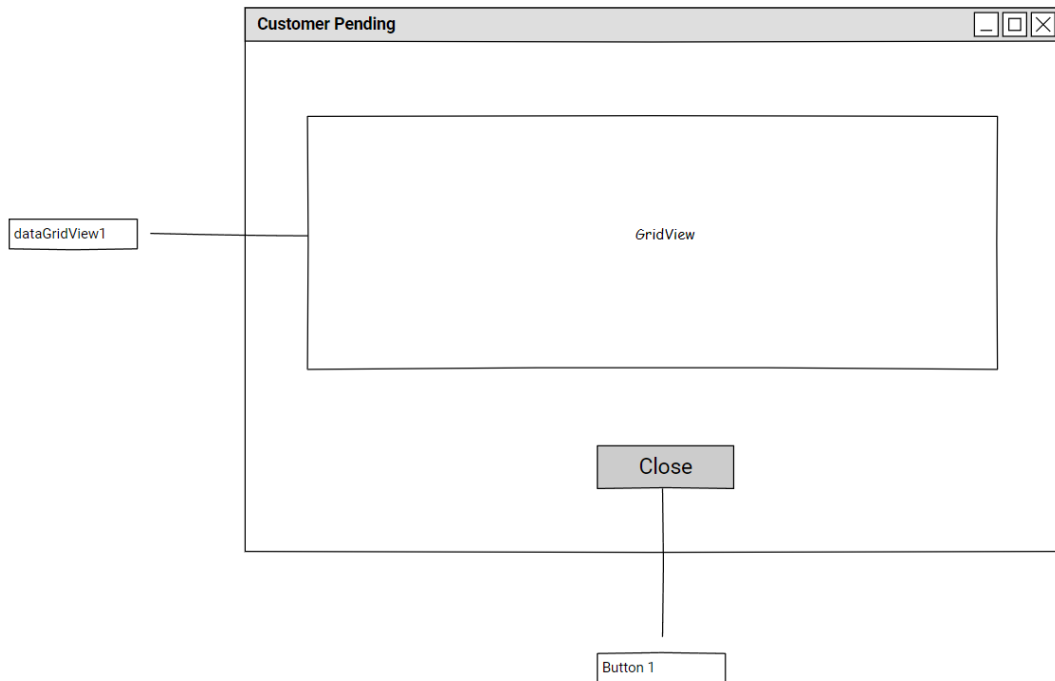
Control	Control Name	Description
Label 1	label1	To apply labels on the relevant controls
Label 2	label2	To apply labels on the relevant controls
Label 3	label3	To apply labels on the relevant controls
Textbox 1	usernametextbox	To allow users to input their username
Textbox 2	passwordtextbox	To allow users to input their password
Button 1	LoginBtn	To enable users to Login into their system



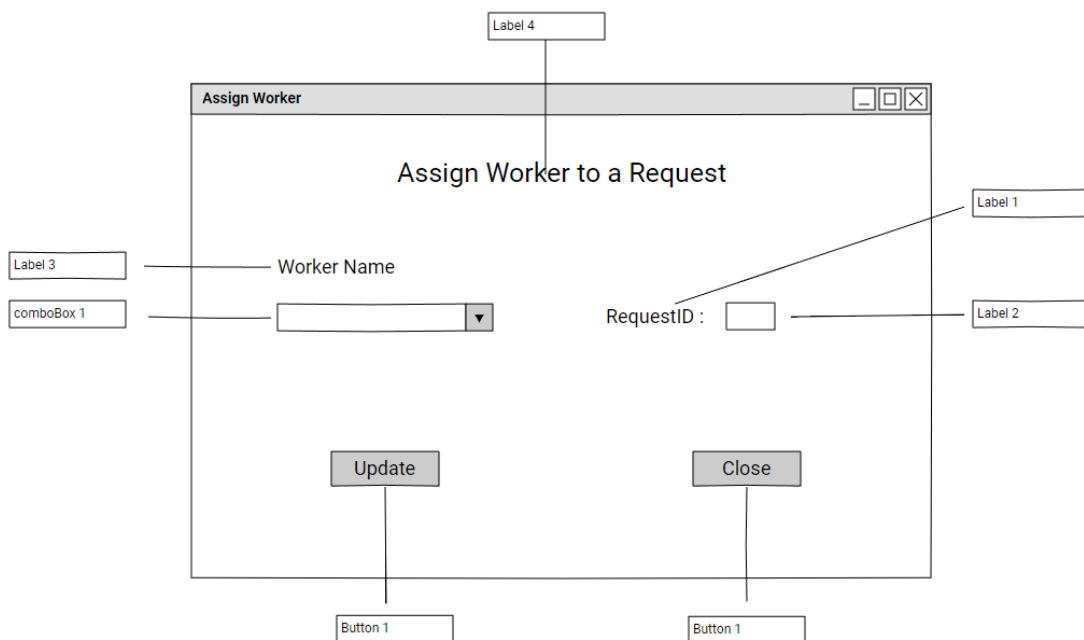
Control	Control Name	Description
Label 1	label1	To apply labels on the relevant controls
Label 2	label2	To apply labels on the relevant controls
Label 3	label3	To apply labels on the relevant controls
Label 5	label5	To apply labels on the relevant controls
Label 6	label6	To apply labels on the relevant controls
Textbox 1	TextBox1	To allow users to input the amount of service
Button 1	CalculateBtn	To enable user to calculate the cost of service
Button 2	NewReqBtn	To send a new request to the worker
Button 3	CloseBtn	To close the screen
listBox1	listBox1	To allow users to choose the type of services
checkBox1	checkBox1	To allow users to choose if the service is urgent



Control	Control Name	Description
Label 1	label1	To apply labels on the relevant controls
dataGridView1	dataGridView1	To display the requested statues
Button 1	SubmitReqBtn	To send a new request to the worker
Button 2	button1	To allow user to check total pending of request
Button 3	CloseBtn	To close the screen

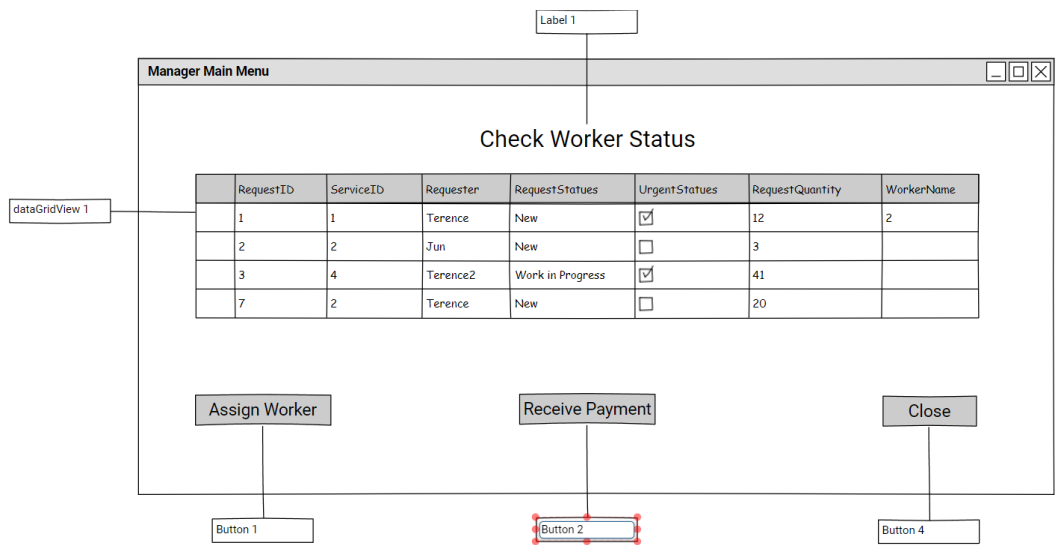


Control	Control Name	Description
dataGridView1	dataGridView1	To display the customer pending request
Button 3	CloseBtn	To close the screen

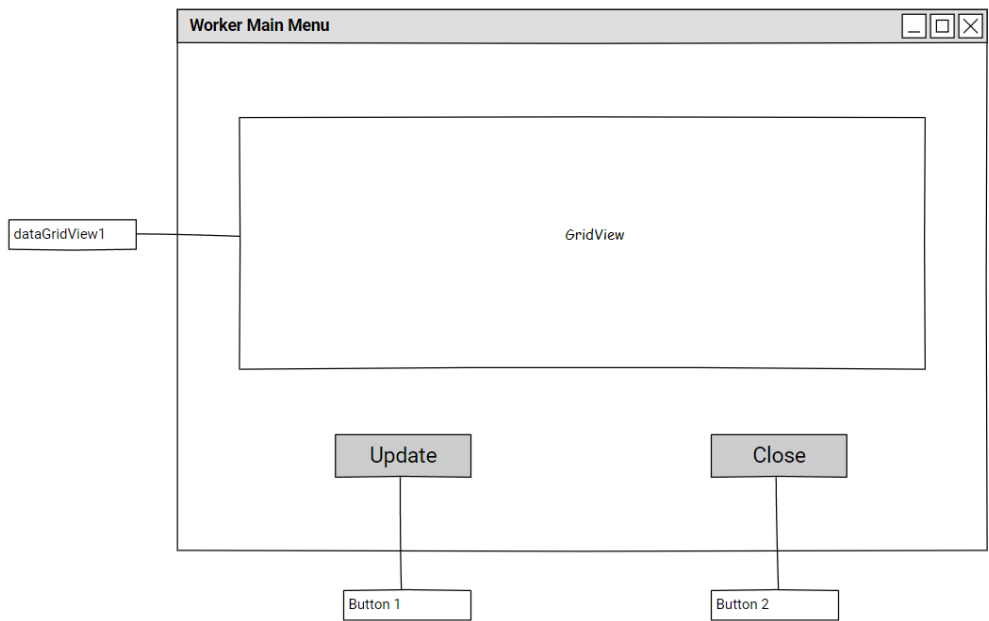


Control	Control Name	Description
Label 1	label1	To apply labels on the relevant controls
Label 2	label2	To show the chosen request ID
Label 3	label3	To apply labels on the relevant controls

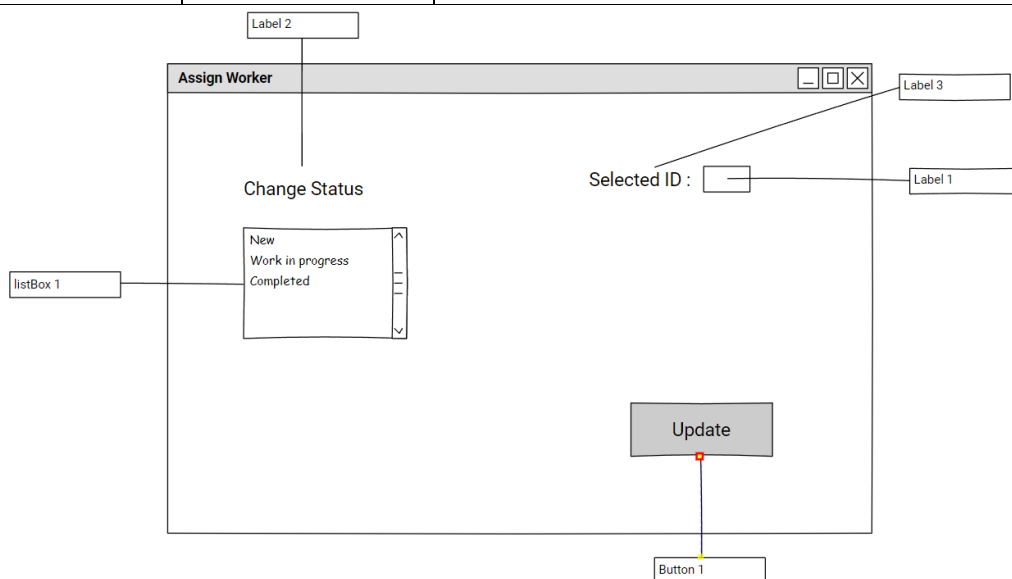
Label 4	label4	To apply labels on the relevant controls
comboBox 1	comboBox1	Let user choose which worker to choose from
Button 1	button3	To update worker of a request
Button 2	button4	To close the screen



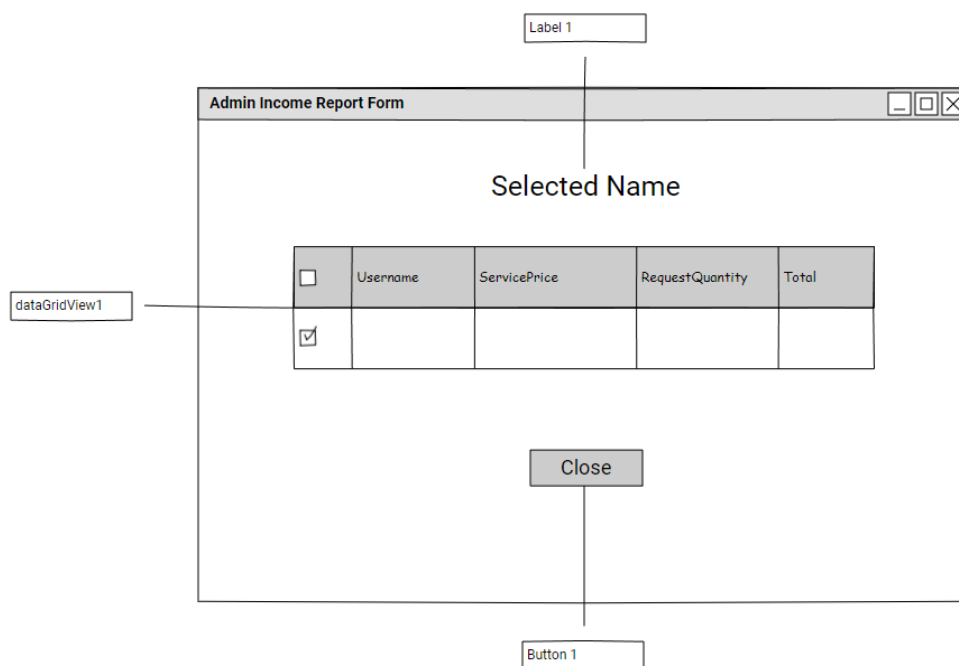
Control	Control Name	Description
Label 1	label1	To apply labels on the relevant controls
dataGridView1	dataGridView1	To display the worker status
Button 1	AssignWorkerBtn	To allow user to assign job to worker
Button 2	ReceivePaymentBtn	Receive payment form customer
Button 4	button4	To close the screen



Control	Control Name	Description
dataGridView1	dataGridView1	To display the customer request status
Button 1	UpdBtn	To update the request status of request
Button 2	CloseBtn	To close the screen

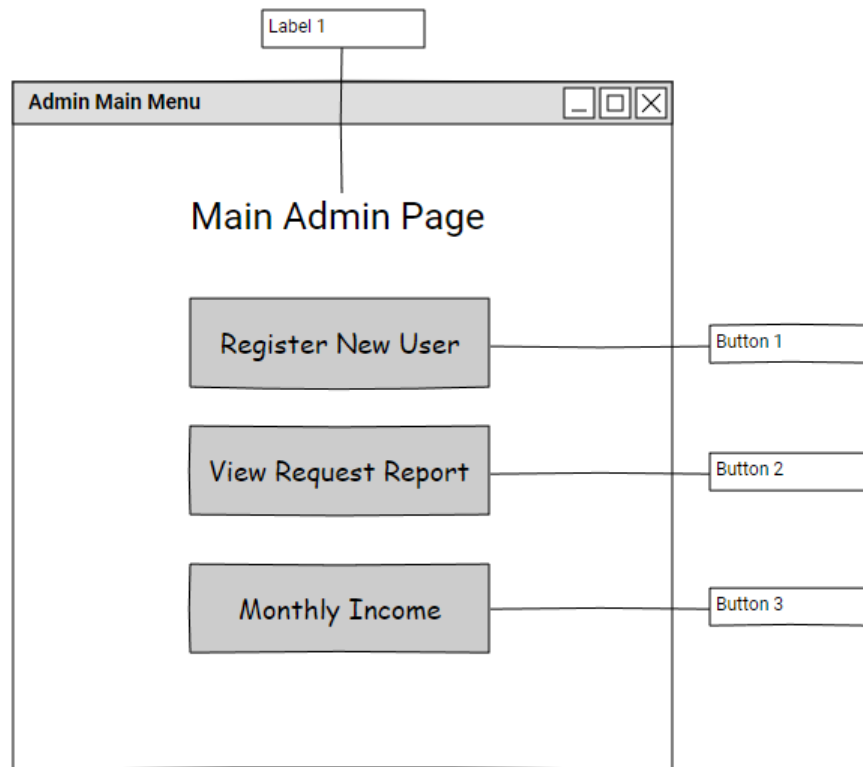


Control	Control Name	Description
Label 1	label1	To show which request ID to change the status
Label 2	label2	To apply labels on the relevant controls
Label 3	label3	To apply labels on the relevant controls
listBox1	listBox1	To allow users to choose the type of request status
Button 1	CalculateBtn	To enable user to calculate the cost of service

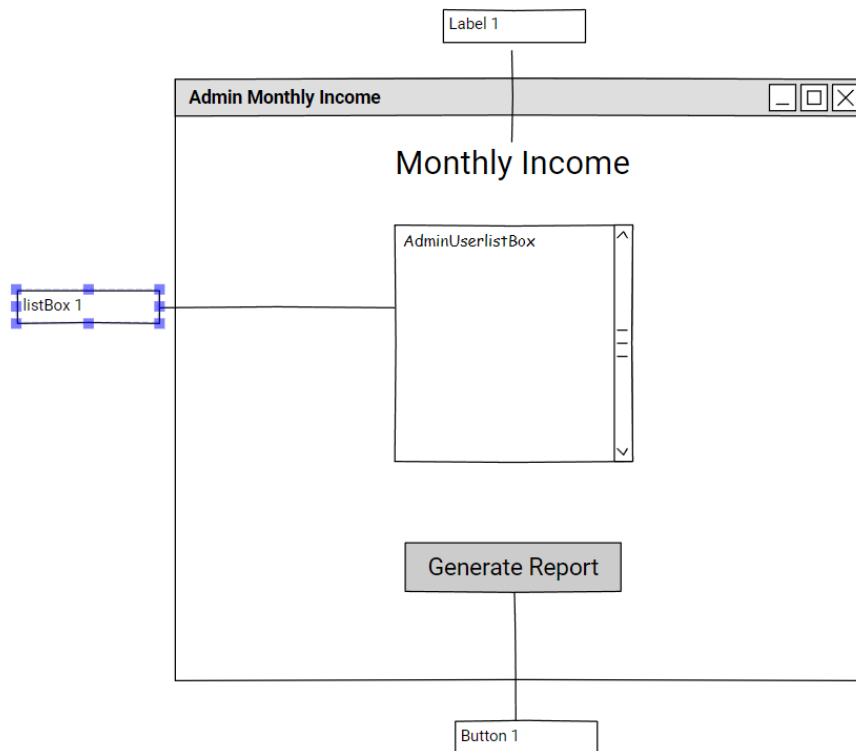


Control	Control Name	Description
---------	--------------	-------------

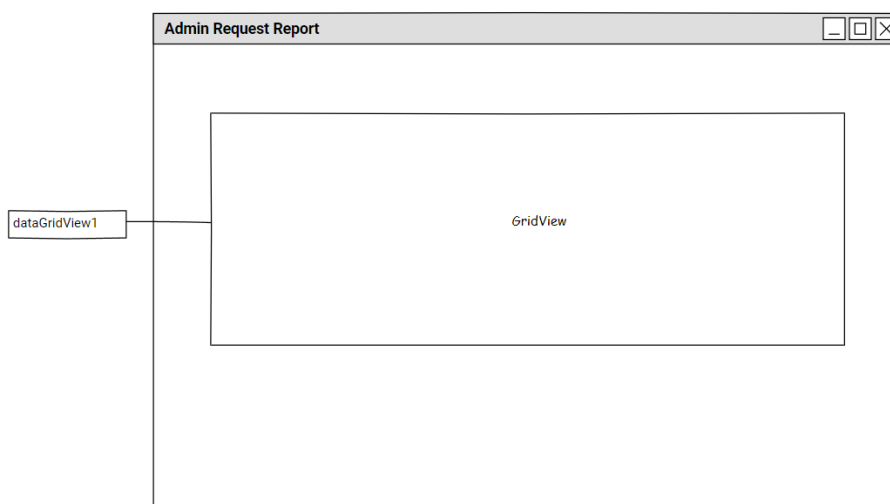
Label 1	UserReportLbl	To apply labels on the relevant controls
dataGridView1	DataGridSQL	To display the selected worker of income report
Button 1	CloseBtn	To close the screen



Control	Control Name	Description
Label 1	label1	To apply labels on the relevant controls
Button 1	Register_form_btn	Direct user to register page
Button 2	Request_form_btn	Direct user to request report page
Button 3	Income_form_btn	Direct user to monthly income page

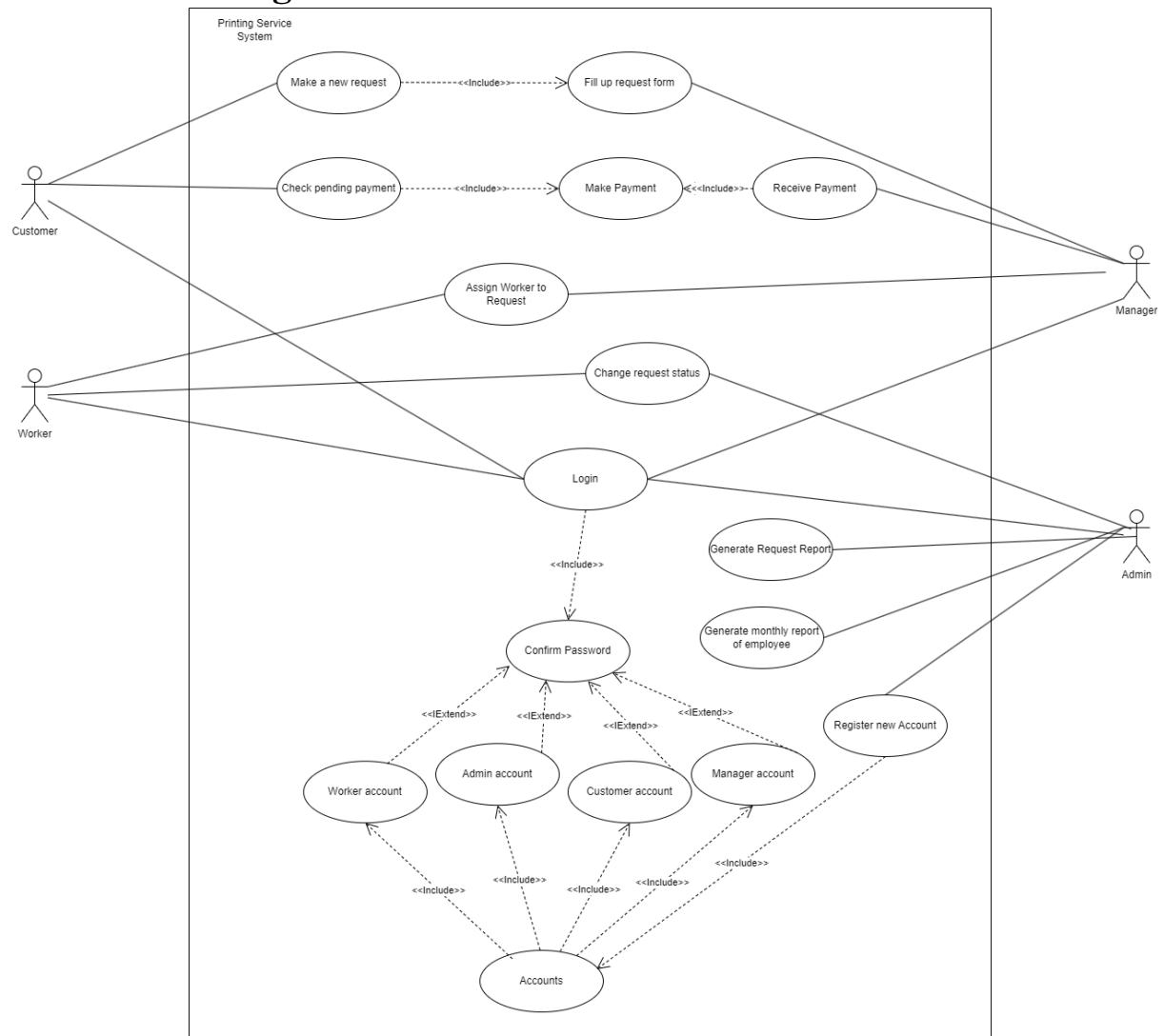


Control	Control Name	Description
Label 1	label1	To apply labels on the relevant controls
listBox 1	AdminUserListBox	To let user choose which account to generate income report
Button 1	GenerateIncomeBtn	To enable user to generate the chosen income

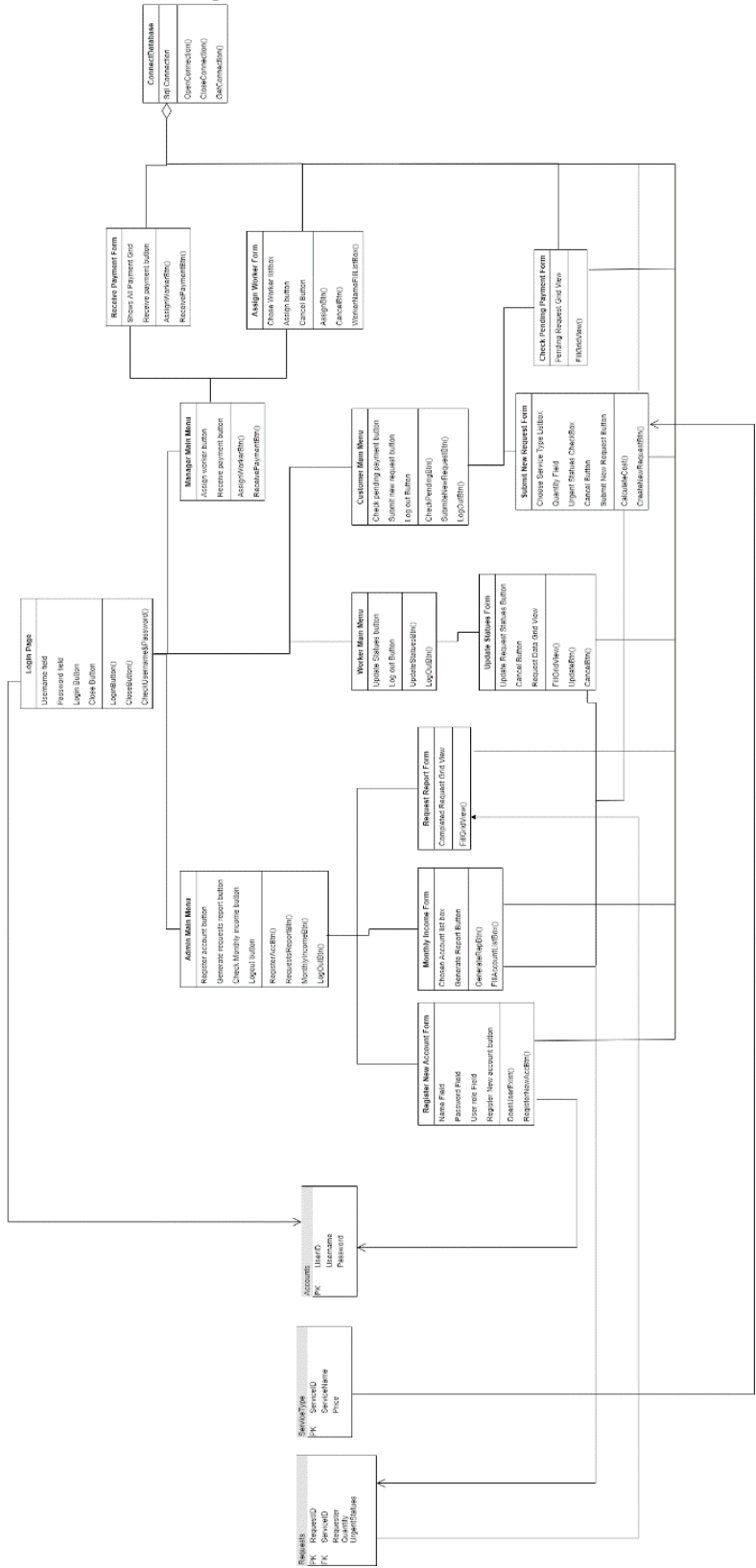


Control	Control Name	Description
dataGridView1	DataGridSQL	To show admin the request report of all accounts

3. Use-case Diagram

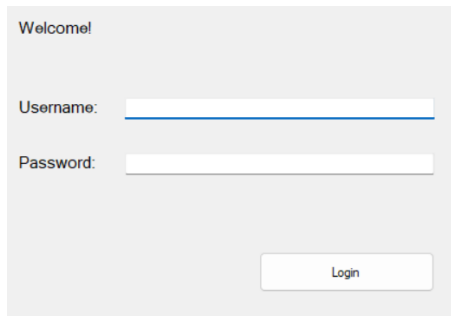


4. Class Diagram



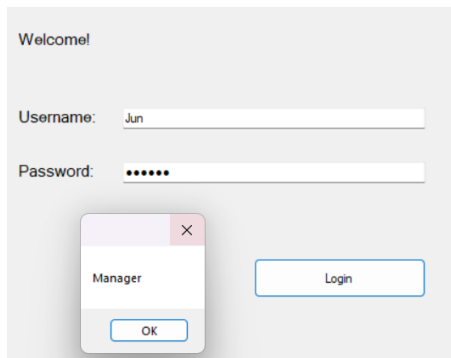
4. Explanation of The implemented codes

Login page



A screenshot of a login page with a light gray background. At the top left, the text "Welcome!" is displayed. Below it, there are two input fields: "Username:" followed by a white text box with a blue underline, and "Password:" followed by a white text box. At the bottom right, there is a white button with the text "Login".

This is the Login page where user will be able to enter their username and password.



A screenshot of the same login page, but now the "Username:" field contains the text "Jun" and the "Password:" field contains six dots. A small dialog box is open in the lower-left area, titled "Manager" with a close button (X) in the top right corner and an "OK" button at the bottom. The "Login" button remains visible on the right.

When user's username and password matched to the database. The program validating its existence in database and it will lets the user into their responding main menu according to their account roles.

```

public partial class LoginPage : Form
{
    SqlConnection conn = new SqlConnection("Data Source=CAESAR;Initial Catalog=IOOPAssignment;Integrated Security=True");
    public static string Username;
    public static string Password;
    1 reference
    public LoginPage()
    {
        InitializeComponent();
    }

    1 reference
    private void LoginBtn_Click(object sender, EventArgs e)
    {
        String username, user_password;

        username = usernametextbox.Text;
        user_password = passwordtextbox.Text;

        String query = "SELECT * FROM ACCOUNTS WHERE Username = '" + username + "' " +
            "AND PASSWORD = '" + user_password + "'";
        SqlCommand cmd = new SqlCommand(query, conn);
        SqlDataAdapter sda = new SqlDataAdapter(query, conn);
        DataTable dt = new DataTable();
        sda.Fill(dt);

        if (dt.Rows.Count > 0)
        {

```

Firstly the program creates an instance of the class SqlConnection with the parameters of the Data Source. Also creating variables Username and Password to override it in further. They are not in any Class method so can be used anywhere inside the Class. There is a Click event of the Login Button where we create the new String variables of username and password and assign values we get from the textboxes of our form. Then we have the SQL query where we use these values. After that we create the objects of Classes SqlCommand, SqlDataAdapter and DataTable. SqlCommand requires 2 parameters: query and connection. In the SqlDataAdapter we can put this object (will do further) to simplify the code and get the data from the Database so we can fill the created table with it.

```

if (dt.Rows.Count > 0)
{
    string roles = (dt.Rows[0][3]).ToString();
    MessageBox.Show(roles);
    MessageBox.Show("Login successful");
    Username = username;
    Password = user_password;
    if (roles == "Customer")
    {
        CustomerMainMenu customerMainMenu = new CustomerMainMenu();
        customerMainMenu.ShowDialog();
    }
    else if (roles == "Manager")
    {
        ManagerMainMenu managerMainMenu = new ManagerMainMenu();
        managerMainMenu.ShowDialog();
    }
    else if (roles == "Admin")
    {
        AdminMainMenu adminMainMenu = new AdminMainMenu();
        adminMainMenu.ShowDialog();
    }
}
else
{

```

There is if statement that checks if the table is not empty where we check the table's first Row and Fourth Column that is Role that we use in other if statements for every role in Accounts table. Depending on the Role we get the program creates an instance of the Class and opens a new Form.

```

else
{
    username = usernametextbox.Text;
    user_password = passwordtextbox.Text;

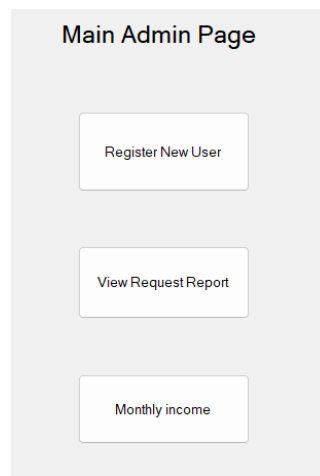
    String query1 = "SELECT * FROM Worker WHERE Username = '" + username + "' " +
        "AND PASSWORD = '" + user_password + "'";
    SqlCommand command = new SqlCommand(query1, conn);
    SqlDataAdapter adapter = new SqlDataAdapter(command);
    DataTable table = new DataTable();
    adapter.Fill(table);

    if (table.Rows.Count > 0)
    {
        MessageBox.Show("Worker");
        MessageBox.Show("Login successful");
        Username = username;
        Password = user_password;
        WorkerMainMenu workerMainManu = new WorkerMainMenu();
        workerMainManu.ShowDialog();
    }
    else
    {
        MessageBox.Show("Incorrect Username or Password");
    }
}
}

```

Here is one more the same piece of code with a new query because we have another table, Worker. So the last Else statement works when the table we created is empty, that means the username or password were incorrect.

Admin Main Page

The image shows a light gray rectangular panel titled "Main Admin Page" at the top. Inside the panel, there are three white rectangular buttons arranged vertically. The top button is labeled "Register New User", the middle button is labeled "View Request Report", and the bottom button is labeled "Monthly income".

This is the form of Admin Main Page where we have 3 buttons that open other forms.

```
public partial class AdminMainMenu : Form
{
    1 reference
    public AdminMainMenu()
    {
        InitializeComponent();
    }

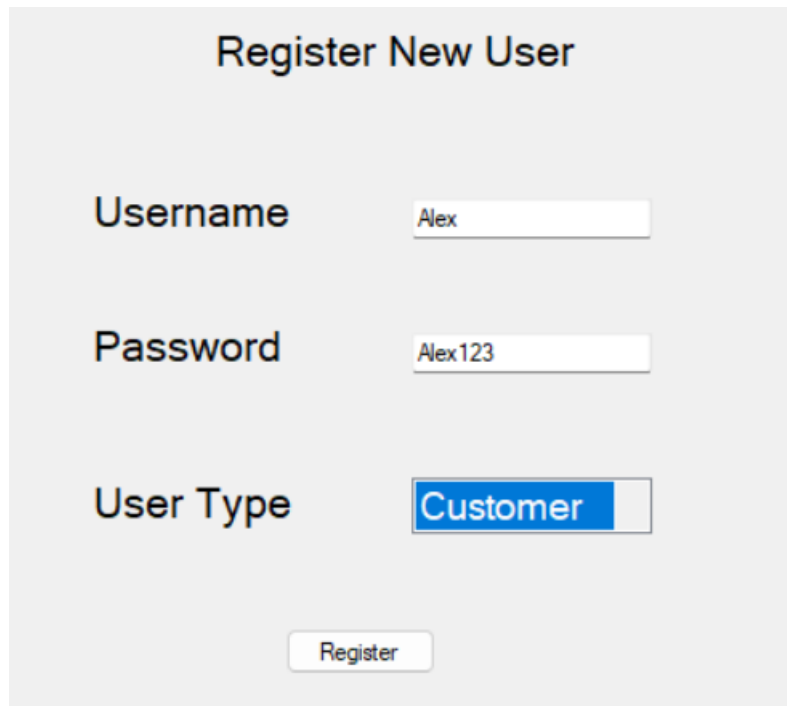
    1 reference
    private void Register_form_btn_Click(object sender, EventArgs e)
    {
        AdminRegisterform regForm = new AdminRegisterform();
        this.Hide();
        regForm.ShowDialog();
        this.Close();
    }

    1 reference
    private void Request_form_btn_Click(object sender, EventArgs e)
    {
        AdminRequestReport requestForm = new AdminRequestReport();
        this.Hide();
        requestForm.ShowDialog();
        this.Close();
    }

    1 reference
    private void Income_form_btn_Click(object sender, EventArgs e)
    {
        AdminMonthlyIncome incomeForm = new AdminMonthlyIncome();
        this.Hide();
        incomeForm.ShowDialog();
        this.Close();
    }
}
```

For all the buttons there are Click events where it connects to another form. When any of the button is clicked, another form will be shown to the user.

Admin Register Form



The image shows a web form titled "Register New User". It has three input fields: "Username" with the value "Alex", "Password" with the value "Alex123", and "User Type" with a dropdown menu showing "Customer". Below these fields is a "Register" button.

This form allows any admin accounts to register a new user account, admin have to enter a unique username, password and assigned a user type to their account

Results		Messages		
	UserID	Username	Password	Role
1	1	Terence	Terence123	Customer
2	2	Jun	Jun123	Manager
3	3	Angelina	Angelina123	Manager
4	4	Sam	Sam123	Admin
5	5	John	John123	Customer

Above shows all the existing account currently in the SQL database

Results		Messages		
	UserID	Username	Password	Role
1	1	Terence	Terence123	Customer
2	2	Jun	Jun123	Manager
3	3	Angelina	Angelina123	Manager
4	4	Sam	Sam123	Admin
5	5	John	John123	Customer
6	6	Alex	Alex123	Customer

After successfully registering, a new user account will be added to the database table.

```
public partial class AdminRegisterform : Form
{
    1 reference
    public AdminRegisterform()
    {
        InitializeComponent();
    }

    3 references
    public Boolean IsUserExists()
    {
        DB dB = new DB();
        DataTable dt = new DataTable();
        SqlDataAdapter adapter = new SqlDataAdapter();
        SqlCommand cmd = new SqlCommand("SELECT * FROM @users WHERE Username = @username", dB.GetConnection());
        cmd.Parameters.Add("@username", SqlDbType.VarChar).Value = NameField.Text;
        if (UserTypeBox.SelectedItem.ToString() == "Customer" || UserTypeBox.SelectedItem.ToString() == "Manager")
            cmd.Parameters.Add("@users", SqlDbType.VarChar).Value = "Accounts";
        else
            cmd.Parameters.Add("@users", SqlDbType.VarChar).Value = "Worker";

        adapter.SelectCommand = cmd;
        adapter.Fill(dt);

        if (dt.Rows.Count > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
```

Here we create the instance of the class DB first to not repeat the same code.

IsUserExists() Method is just to keep all the usernames unique.

```
40 references
class DB
{
    SqlConnection connection = new SqlConnection("Data Source=CAESAR;Initial Catalog=IOO")

    6 references
    public void OpenConnection()
    {
        if (connection.State == System.Data.ConnectionState.Closed)
            connection.Open();
    }

    6 references
    public void CloseConnection()
    {
        if (connection.State == System.Data.ConnectionState.Open)
            connection.Close();
    }

    20 references
    public SqlConnection GetConnection()
    {
        return connection;
    }
}
```

Above shows the Class DB we created that has 3 methods which allows the connection to be easily open and close.

```
private void RegisterBtn_Click(object sender, EventArgs e)
{
    if (NameField.Text == "Enter your name..." || NameField.Text == "")
        return;
    if (UserTypeBox.SelectedItem.ToString() == "Customer" || UserTypeBox.SelectedItem.ToString() == "Manager")
    {
        if (IsUserExists())
        {
            MessageBox.Show("Account with this username already exists");
            return;
        }
        DB db = new DB();
        SqlCommand cmd = new SqlCommand("INSERT INTO Accounts (Username, Password, Role) VALUES (@username, @password, @role)", db.GetCo
        cmd.Parameters.Add("@username", SqlDbType.VarChar).Value = NameField.Text;
        cmd.Parameters.Add("@password", SqlDbType.VarChar).Value = PasswordField.Text;
        cmd.Parameters.Add("@role", SqlDbType.VarChar).Value = UserTypeBox.SelectedItem.ToString();
        db.OpenConnection();

        if (cmd.ExecuteNonQuery() == 1)
            MessageBox.Show("Account registered");
        else
            MessageBox.Show("Account haven't registered");
        db.CloseConnection();
    }
    else
    {

```

This is the Button Click event where we call the method IsUserExists() and if statement that checks if the user role is Customer or Manager. We have the query to insert the data from the textboxes and listbox. With the DB Class methods OpenConnection() and CloseConnection() we provide the changes to the database.

```
else
{
    if (IsUserExists())
    {
        MessageBox.Show("Account with this username already exists");
        return;
    }
    DB db = new DB();
    SqlCommand cmd = new SqlCommand("INSERT INTO Worker (Username, Password) VALUES (@username, @password)", db.GetConnection());
    cmd.Parameters.Add("@username", SqlDbType.VarChar).Value = NameField.Text;
    cmd.Parameters.Add("@password", SqlDbType.VarChar).Value = PasswordField.Text;

    db.OpenConnection();
    if (cmd.ExecuteNonQuery() == 1)
        MessageBox.Show("Account registered");
    else
        MessageBox.Show("Account haven't registered");
    db.CloseConnection();

    if (IsUserExists())
        return;
}
}
```

The IsUserExists() methods is also called when inserting new data into the Worker Table.

Admin View Request Report

	UserID	Usemame	Password	Role	Request
▶	1	Terence	Terence123	Customer	1
	1	Terence	Terence123	Customer	4
*					

Above shows the DataGridView that shows all the request completed by the workers.

```
public partial class AdminRequestReport : Form
{
    1 reference
    public AdminRequestReport()
    {
        InitializeComponent();
        LoadData();
    }
    1 reference
    private void LoadData()
    {
        string query = "SELECT * FROM Accounts a JOIN Request r ON a.UserID = r.Worker WHERE r.RequestStatues = 'Completed'";

        DB db = new DB();
        SqlCommand command = new SqlCommand(query, db.GetConnection());

        SqlDataAdapter adapter = new SqlDataAdapter(command);
        DataTable table = new DataTable();
        adapter.Fill(table);

        DataGridViewSQL.DataSource = table;
    }
}
```

In this form we load the data for the DataGridView using the query where Request Status is completed. It will fill the DataTable and override the DataGridViewSQL.

Admin Monthly Income Report



A screenshot of a Windows application window titled "Admin Monthly Income Report". Inside the window, there is a list box in the top-left corner containing three items: "Ame", "John", and "Bob". Below the list box, centered at the bottom of the window, is a button labeled "Generate report".

In this form admin will be able to see all existing workers in the database and select a worker to generate his/her income report.

```
public void FillUserList()
{
    DB db = new DB();
    DataTable dt = new DataTable();
    SqlCommand cmd = new SqlCommand("SELECT Username FROM Worker", db.GetConnection());
    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    adapter.Fill(dt);
    foreach (DataRow dr in dt.Rows)
    {
        AdminUserListBox.Items.Add(dr[0].ToString());
    }
}

1 reference
private void GenerateIncomeBtn_Click(object sender, EventArgs e)
{
    if (AdminUserListBox.SelectedItem != null)
    {
        AdminIncomeReportForm incomeForm = new AdminIncomeReportForm();
        incomeForm.UserReportLabelText = AdminUserListBox.SelectedItem.ToString();
        this.Hide();
        incomeForm.ShowDialog();
        this.Close();
    }
    else
    {
        MessageBox.Show("Please choose a user");
    }
}
```

Firstly the table is filled with the data from the database and with the foreach loop add them to the listbox. Generate Income Button will opens an Income Form and overrides the value from Selected Item to UserReportLabelText.

Ame

	Username	ServicePrice	RequestQuantity	Total
▶	Ame	0,80	10	8,00
	Ame	5,00	30	150,00
*				

Total Income 158

Close

If we choose user Ame we can see all his completed requests and the system will calculate his total income from his requests.

```

public AdminIncomeReportForm()
{
    InitializeComponent();
}

1 reference
public string UserReportLabelText
{
    get { return UserReportLbl.Text; }
    set
    {
        UserReportLbl.Text = value;
        DataGridViewFill();
    }
}

1 reference
public int GetUserIDByName(string username)
{
    int userID = 0;
    DB db = new DB();
    DataTable dt = new DataTable();
    SqlCommand cmd = new SqlCommand("SELECT UserID FROM Worker WHERE UserName = @username", db.Get
    cmd.Parameters.Add("@username", SqlDbType.VarChar).Value = username;
    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    adapter.Fill(dt);

    if (dt.Rows.Count > 0)
    {
        userID = (int)dt.Rows[0][0];
    }

    return userID;
}

```

After the initializing the form the UserReportLbl gets the value from previous form where Admin choose the user when we clicked the Generate report button.

Customer Main Menu

Request Statues

	ServiceName	RequestStatues	UrgentStatues	RequestQuantity	Username	ServicePrice	Total
▶	Printing A4 - Blac...	Completed	<input type="checkbox"/>	10	Terence	0,80	8,00
	Binding - Comb Bi...	Completed	<input checked="" type="checkbox"/>	30	Terence	5,00	150,00
*			<input type="checkbox"/>				

Submit new requests Check total pending Close

This form contains a DataGridView with all the customer current requests. There are buttons for 2 other forms. Which allows customer to make a new request

```
1 reference
public CustomerMainMenu()
{
    InitializeComponent();
    LoadData();
}

1 reference
private void LoadData()
{
    string username = LoginPage.Username;
    int userID = GetUserIDByName(username);
    string query = @"SELECT s.ServiceName, r.RequestStatues, r.UrgentStatues,
        r.RequestQuantity, a.Username, s.ServicePrice, (s.ServicePrice * r.RequestQuantity) as Total
    FROM Request r JOIN Accounts a ON r.Worker = a.userID JOIN ServiceType s ON
        r.ServiceID = s.ServiceID WHERE a.UserID = @userID";

    DB db = new DB();
    SqlCommand command = new SqlCommand(query, db.GetConnection());
    command.Parameters.AddWithValue("@userID", userID);

    SqlDataAdapter adapter = new SqlDataAdapter(command);
    DataTable table = new DataTable();
    adapter.Fill(table);

    CustDataGrid.DataSource = table;
}
```

With the LoadData() methods, we can get selected columns using objects of other classes and put them inside a data table to show it in the Data Grid.


```
1 reference
private void CloseBtn_Click(object sender, EventArgs e)
{
    this.Close();
}

1 reference
private void SubmitReqBtn_Click(object sender, EventArgs e)
{
    CustNewRequestForm custNewRequestForm = new CustNewRequestForm();
    custNewRequestForm.ShowDialog();
}

1 reference
private void button1_Click(object sender, EventArgs e)
{
    CustomerPending customerPending = new CustomerPending();
    customerPending.ShowDialog();
}
```

The event for buttons, when user perform a click action on these button, the system will show the user responding form.

Customer Pending

	RequestID	ServiceName	RequestStatues	UrgentStatues	RequestQuantity	ServicePrice
▶	8	Binding - Think C...	New	<input type="checkbox"/>	20	15.00
*				<input type="checkbox"/>		

Pay

Close

In this form, the Data Grid will shows the customer which of their request are still pending for payment. There is a button “Pay” which will redirect customer to another form for them to make payment

```
public partial class CustomerPending : Form
{
    1 reference
    public CustomerPending()
    {
        InitializeComponent();
        CustomerPending_Load();
    }

    1 reference
    private void CustomerPending_Load()
    {
        DB dB = new DB();
        string query = @"SELECT r.RequestID, s.ServiceName, r.RequestStatues, r.UrgentStatues, r.RequestQuantity,
                           s.ServicePrice, (r.RequestQuantity * s.ServicePrice) as Total, r.PaymentStatus
                           FROM Request r JOIN ServiceType s ON r.ServiceID = s.ServiceID
                           WHERE r.Requester = '{LoginPage.Username}' AND NOT r.PaymentStatus = 'Paid'";

        SqlCommand cmd = new SqlCommand(query, dB.GetConnection());
        SqlDataAdapter sda = new SqlDataAdapter(cmd);
        DataTable dt = new DataTable();
        sda.Fill(dt);
        DataGridView1.DataSource = dt;
    }

    1 reference
    private void CloseBtn_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
```

```

1 reference
private void CloseBtn_Click(object sender, EventArgs e)
{
    this.Close();
}

1 reference
private void PayBtn_Click(object sender, EventArgs e)
{
    DataGridViewRow selectedRow = DataGridSQL.SelectedRows[0];
    int requestID = Convert.ToInt32(selectedRow.Cells["RequestID"].Value);
    ManagerReceivePayment receivePayment = new ManagerReceivePayment(requestID);
    this.Hide();
    receivePayment.ShowDialog();
    this.Close();
}

```

There is two button, the pay button which redirect customer to the pay form to allow them to pay for their pending request.

Customer New Request

Service type	Fees (RM)	Amount: 55
Printing A4 - Black and white	0.8 / page	<input checked="" type="checkbox"/> Urgent (Will be prioritized however there is a subcharge of 30%) Cost: 357,5
Printing A4 - Color	2.50 / page	
Binding - Comb Binding	5 / page	
Binding - Thick Cover	15 / book	
Poster printing (A0 - A3)	3 / page	
Banner	10 / banner	

A4 - BW
 A4 - Color
Binding - CB

Calculate

Submit new Request

Close

This is the form where customer can select which service they require, type in the amount. There is also an option for urgent request where their request will be prioritize, however there will be a sub charge of 30%. Before submitting the request the system will calculate the total cost, after showing user will be able to submit their new request.

	RequestID	ServiceID	Requester	RequestStatus	UrgentStatus	RequestQuantity	Worker	PaymentStatus
1	1	1	Lyra	Completed	NULL	10	1	NULL
2	2	2	Terence	New	1	25	3	Paid
3	3	2	Terence	New	0	25	2	NULL
4	4	3	John	Completed	1	30	1	NULL
5	6	1	John	New	1	55	3	Not paid
6	7	5	terence	New	0	15	NULL	Paid
7	8	4	Terence	New	0	20	NULL	Not paid

This is the SQL table for the existing request in the Request database

Results Messages								
	RequestID	ServiceID	Requester	RequestStatus	UrgentStatus	RequestQuantity	Worker	PaymentStatus
1	1	1	Lyra	Completed	NULL	10	1	NULL
2	2	2	Terence	New	1	25	3	Paid
3	3	2	Terence	New	0	25	2	NULL
4	4	3	John	Completed	1	30	1	NULL
5	6	1	John	New	1	55	3	Not paid
6	7	5	terence	New	0	15	NULL	Paid
7	8	4	Terence	New	0	20	NULL	Not paid
8	9	3	Terence	New	1	55	NULL	Not paid

After adding the request a new request can be seen in the bottom of the list.

```

1 reference
public CustNewRequestForm()
{
    InitializeComponent();
}

1 reference
private void CustNewRequestForm_Load(object sender, EventArgs e)
{
    NewReqBtn.Visible = false;
    listBox1.SelectedIndex = 1;
}

1 reference
private void CalculateBtn_Click(object sender, EventArgs e)
{
    double totalCost;
    double ServiceCost = 0;
    int Quantity;
    double Subcharged = 1;

    if (listBox1.SelectedItem.ToString() == "A4 - BW")
    {
        ServiceCost = 0.8;
    }
    else if (listBox1.SelectedItem.ToString() == "A4 - Color")
    {
        ServiceCost = 2.5;
    }
}

```

NewReqBtn.Visible = false makes the Submit Request button is invisible till the user calculates the total amount. And here is the CalculateBtn Click event to calculate the total amount on customer choice.

```

        ServiceCost = 0.8;
    }
    else if(listBox1.SelectedItem.ToString() == "A4 - Color")
    {
        ServiceCost = 2.5;
    }
    else if (listBox1.SelectedItem.ToString() == "Binding - CB")
    {
        ServiceCost = 5.0;
    }
    else if (listBox1.SelectedItem.ToString() == "Binding - TC")
    {
        ServiceCost = 15.0;
    }
    else if (listBox1.SelectedItem.ToString() == "Poster")
    {
        ServiceCost = 3.0;
    }
    else if (listBox1.SelectedItem.ToString() == "Banner")
    {
        ServiceCost = 10.0;
    }

    if (checkBox1.Checked)
    {
        Subcharged = 1.3;
    }

    Quantity = int.Parse(AmountBox.Text);
    totalCost = ServiceCost * Quantity * Subcharged;
    CostLbl.Text = totalCost.ToString();
    NewReqBtn.Visible = true;
}

```

After the “if” statements depending on the selected item in listbox and if request is urgent the program calculates the total amount and makes Submit Request Button visible.

```

private void NewReqBtn_Click(object sender, EventArgs e)
{
    int urgent = 0;
    int Quantity = int.Parse(AmountBox.Text);
    int ServiceType = 0;
    if (checkBox1.Checked)
    {
        urgent = 1;
    }

    if (listBox1.SelectedItem.ToString() == "A4 - BW")
    {
        ServiceType = 1;
    }
    else if (listBox1.SelectedItem.ToString() == "A4 - Color")
    {
        ServiceType = 2;
    }
    else if (listBox1.SelectedItem.ToString() == "Binding - CB")
    {
        ServiceType = 3;
    }
    else if (listBox1.SelectedItem.ToString() == "Binding - TC")
    {
        ServiceType = 4;
    }
    else if (listBox1.SelectedItem.ToString() == "Poster")
    {
        ServiceType = 5;
    }
}

```

```

    }
    else if (listBox1.SelectedItem.ToString() == "Poster")
    {
        ServiceType = 5;
    }
    else if (listBox1.SelectedItem.ToString() == "Banner")
    {
        ServiceType = 6;
    }

    DB db = new DB();

    string query = "Insert into Request values (@ServiceID, @Name, @Statues, @Urgent, @Quantity, @Worker, @Payment) ";
    SqlCommand cmd = new SqlCommand(query, db.GetConnection());
    cmd.Parameters.AddWithValue("@ServiceID", ServiceType);
    cmd.Parameters.AddWithValue("@Name", LoginPage.Username);
    cmd.Parameters.AddWithValue("@Statues", "New");
    cmd.Parameters.AddWithValue("@Urgent", urgent);
    cmd.Parameters.AddWithValue("@Quantity", Quantity);
    cmd.Parameters.AddWithValue("@Payment", "Not paid");
    cmd.Parameters.AddWithValue("@Worker", DBNull.Value);
    db.OpenConnection();
    cmd.ExecuteNonQuery();
    db.CloseConnection();
    MessageBox.Show("Request added");
    this.Close();
}
}

```

Submit Button Click event that get all the data from customer and insert it into a database Request table. Payment status will always be not paid until customer pays for the request. And the Worker Column will be null until Manager assigns the worker for the request.

Manager Main Menu

Check Work Status							
	RequestID	ServiceName	Requester	RequestStatus	UrgentStatus	RequestQuantity	ServicePrice
▶	2	Printing A4 - Color	Terence	New	<input checked="" type="checkbox"/>	25	2,50
	3	Printing A4 - Color	Terence	New	<input type="checkbox"/>	25	2,50
	6	Printing A4 - Blac...	John	New	<input checked="" type="checkbox"/>	55	0,80
	7	Poster printing(A0...	terence	New	<input type="checkbox"/>	15	3,00
	8	Binding - Think C...	Terence	New	<input type="checkbox"/>	20	15,00
	9	Binding - Comb Bl...	Terence	New	<input checked="" type="checkbox"/>	55	5,00
*					<input type="checkbox"/>		

Assign Worker

Receive Payment

Close

The DataGridView that shows all the requests that are not Completed.

Check Work Status								
	Requester	RequestStatus	UrgentStatus	RequestQuantity	ServicePrice	Total	PaymentStatus	W
▶	nce	New	<input checked="" type="checkbox"/>	25	2,50	62,50	Paid	3
	nce	New	<input type="checkbox"/>	25	2,50	62,50	Paid	2
		New	<input checked="" type="checkbox"/>	55	0,80	44,00	Not paid	3
	nce	New	<input type="checkbox"/>	20	15,00		paid	
	nce	New	<input checked="" type="checkbox"/>	55	5,00		paid	
*			<input type="checkbox"/>					

Assign Worker

Receive Payment

Close

This request is already paid

OK

Manager cannot receive payment for already paid requests.


```

public partial class ManagerMainMenu : Form
{
    1 reference
    public ManagerMainMenu()
    {
        InitializeComponent();
        LoadData();
    }

    1 reference
    private void LoadData()
    {
        string query = @"SELECT r.RequestID, s.ServiceName, r.Requester, r.RequestStatuses, r.UrgentStatuses, r.RequestQuantity,
                           s.ServicePrice, (r.RequestQuantity * s.ServicePrice) as Total,
                           r.PaymentStatus, r.Worker FROM Request r JOIN ServiceType s
                           ON r.ServiceID = s.ServiceID WHERE NOT r.RequestStatuses = 'Completed'";

        DB db = new DB();
        SqlCommand command = new SqlCommand(query, db.GetConnection());

        SqlDataAdapter adapter = new SqlDataAdapter(command);
        DataTable table = new DataTable();
        adapter.Fill(table);

        DataGridSQL.DataSource = table;
    }

    0 references
    private void button4_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}

```

```

1 reference
private void AssignWorkerBtn_MouseClick(object sender, MouseEventArgs e)
{
    DataGridViewRow selectedRow = DataGridSQL.SelectedRows[0];
    int requestID = Convert.ToInt32(selectedRow.Cells["RequestID"].Value);
    ManagerAssignWorker assignWorker = new ManagerAssignWorker(requestID);
    this.Hide();
    assignWorker.ShowDialog();
    this.Close();
}

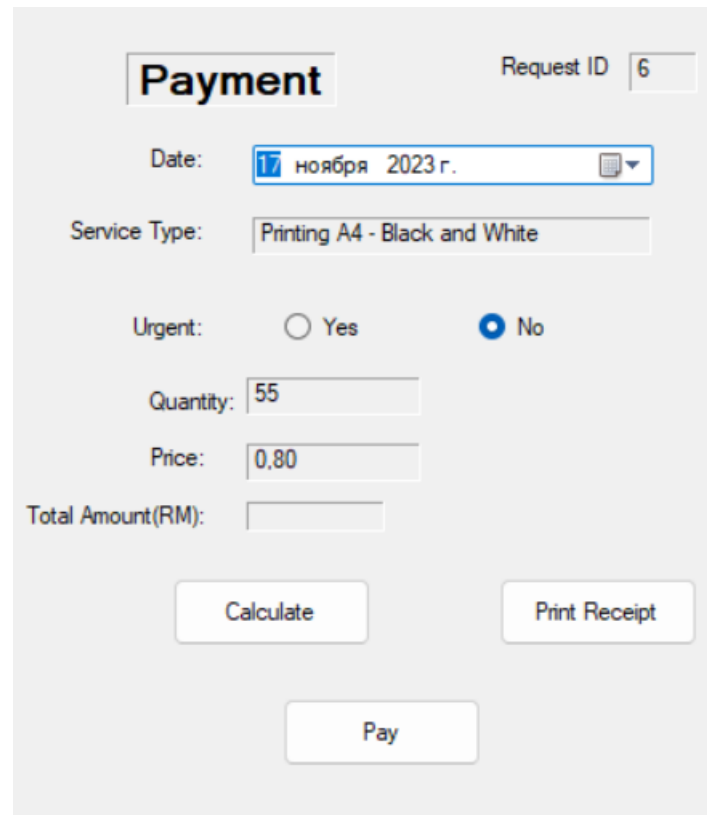
1 reference
private void ReceivePaymentBtn_Click(object sender, EventArgs e)
{
    DataGridViewRow selectedRow = DataGridSQL.SelectedRows[0];
    int requestID = Convert.ToInt32(selectedRow.Cells["RequestID"].Value);
    string payment_status = selectedRow.Cells["PaymentStatus"].Value.ToString();
    if (payment_status != "Paid")
    {
        ManagerReceivePayment receivePayment = new ManagerReceivePayment(requestID);
        this.Hide();
        receivePayment.ShowDialog();
        this.Close();
    }
    else
    {
        MessageBox.Show("This request is already paid");
        return;
    }
}
}

```

In Assign Worker and Receive Payment Button events there are some variables that get overrode from the Data Table and send values to Classes of forms which opens when the buttons get clicked.

In Receive Payment Button Click event there is a “IF” statement that checks if Payment Status is Paid. If NOT paid the form opens, if not – manager gets the message.

Receive Payment (Also Customer's Pay button)



The image shows a web-based payment form titled "Payment". At the top right, there is a "Request ID" field with the value "6". Below the title, there is a "Date" field showing "17 ноября 2023 г." with a calendar icon. The "Service Type" is set to "Printing A4 - Black and White". Under the "Urgent" section, there are two radio buttons: "Yes" (unselected) and "No" (selected). The "Quantity" field contains the value "55", and the "Price" field contains "0,80". The "Total Amount(RM)" field is empty. At the bottom, there are three buttons: "Calculate", "Print Receipt", and "Pay".

Field	Value
Request ID	6
Date	17 ноября 2023 г.
Service Type	Printing A4 - Black and White
Urgent	No
Quantity	55
Price	0,80
Total Amount(RM)	

In this form the program will get the request detail such as ServiceType, Urgent Status, Quantity and Price from the Database. After that the customer can pay for their pending request.

```

6 references
public partial class ManagerReceivePayment : Form
{
    string urgent_status = "";
    2 references
    public ManagerReceivePayment(int requestID)
    {
        InitializeComponent();
        RequestID.Text = requestID.ToString();
        LoadData();
    }

    1 reference
    private void LoadData()
    {
        string query = @"SELECT s.ServicePrice, s.ServiceName, r.RequestQuantity, r.UrgentStatues FROM ServiceType s
        JOIN Request r ON s.ServiceID = r.ServiceID
        WHERE r.RequestID = '{RequestID.Text}'";

        DB db = new DB();
        SqlCommand command = new SqlCommand(query, db.GetConnection());

        SqlDataAdapter adapter = new SqlDataAdapter(command);
        DataTable table = new DataTable();
        adapter.Fill(table);

        PriceLbl.Text = table.Rows[0][0].ToString();
        ServiceTypeLbl.Text = table.Rows[0][1].ToString();
        QuantityLbl.Text = table.Rows[0][2].ToString();
        urgent_status = table.Rows[0][3].ToString();
    }
}

```

Using table.Rows[x][y] program gets the values of x rows and y columns.

```

PriceLbl.Text = table.Rows[0][0].ToString();
ServiceTypeLbl.Text = table.Rows[0][1].ToString();
QuantityLbl.Text = table.Rows[0][2].ToString();
urgent_status = table.Rows[0][3].ToString();

if (urgent_status == "1")
{
    RadioBtnYes.Checked = true;
}
else
{
    RadioBtnNo.Checked = true;
}
}

1 reference
private void CalculateBtn_Click(object sender, EventArgs e)
{
    double price = double.Parse(PriceLbl.Text);
    int quantity = Convert.ToInt32(QuantityLbl.Text);
    double total_amount = price * quantity;

    if (RadioBtnYes.Checked)
    {
        total_amount *= 1.3; // Increase total amount by 30%
    }

    TotalAmount.Text = "RM" + total_amount.ToString();
}

1 reference
private void PrintReceiptBtn_Click(object sender, EventArgs e)
{
    MessageBox.Show("Receipt printed!");
}

```

```

1 reference
private void FinishBtn_Click(object sender, EventArgs e)
{
    string query = $"UPDATE Request SET PaymentStatus = 'Paid', UrgentStatuses = @status WHERE RequestID = '{RequestID.Text}'";
    DB db = new DB();
    SqlCommand command = new SqlCommand(query, db.GetConnection());

    if (RadioBtnYes.Checked)
        command.Parameters.AddWithValue("@status", "1");
    else
        command.Parameters.AddWithValue("@status", "0");

    SqlDataAdapter adapter = new SqlDataAdapter(command);
    DataTable table = new DataTable();
    adapter.Fill(table);
    db.OpenConnection();
    if (command.ExecuteNonQuery() == 1)
        MessageBox.Show("Request updated");
    else
        MessageBox.Show("Request haven't been updated");
    db.CloseConnection();
    this.Close();
}

```

If statement to check which radio button checked to update the Urgent Status.

RequestID	ServiceID	Requester	RequestStatuses	UrgentStatuses	RequestQuantity	Worker	PaymentStatus
6	1	John	New	1	55	3	Not paid

Table before paying and Urgent Status was 1. Which indicated that the customer has not pay for the request yet.

Payment

Request ID 6

Date: 17 ноября 2023 г.

Service Type: Printing A4 - Black and White

Urgent: ☐ Yes ☒ No

Quantity: 55

Price: 0,80

Total Amount(RM): RM44

Calculate

Request updated

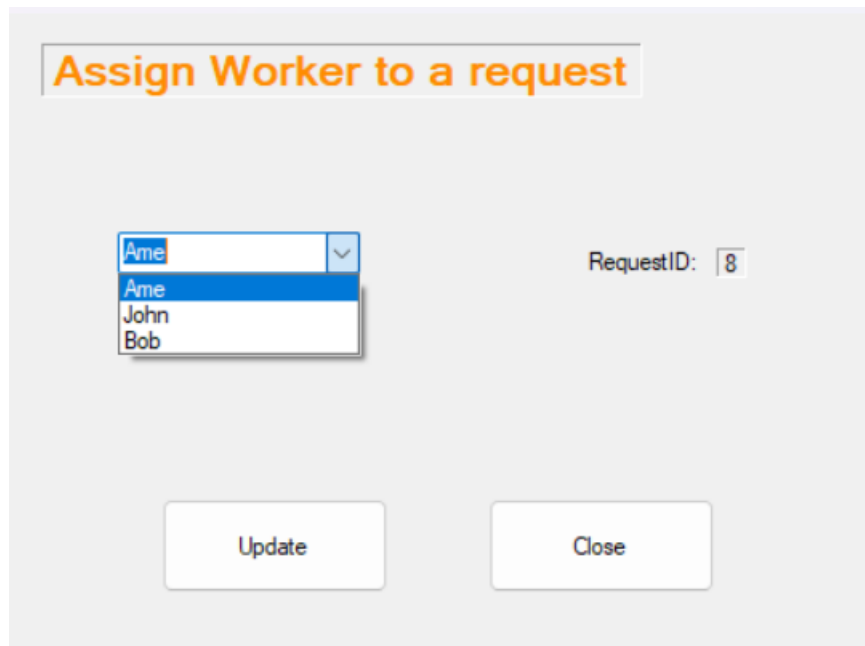
OK

Pay

RequestID	ServiceID	Requester	RequestStatues	UrgentStatues	RequestQuantity	Worker	PaymentStatus
6	1	John	New	0	55	3	Paid

After updating the request on the form, the data in the SQL table will reflect the changes and change the payment status to Paid

Manager Assign Worker



Assign Worker to a request

Worker: Ame
RequestID: 8

Update Close

In this form, Manager can choose which worker to assign to the request, when the update button is clicked the request with the ID shown will be updated.

RequestID	ServiceID	Requester	RequestStatues	UrgentStatues	RequestQuantity	Worker	PaymentStatus
8	4	Terence	New	0	20	NULL	Not paid

Above is the existing request existing where the worker have NULL value

RequestID	ServiceID	Requester	RequestStatues	UrgentStatues	RequestQuantity	Worker	PaymentStatus
8	4	Terence	New	0	20	2	Not paid

After Updating, the request has been assigned a new worker.

```

public partial class ManagerAssignWorker : Form
{
    1 reference
    public ManagerAssignWorker(int requestID)
    {
        InitializeComponent();
        RequestID.Text = requestID.ToString();
        FillUsernameBox();
    }

    1 reference
    private void button4_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    1 reference
    public int GetUserIDByName(string username)
    {
        int userID = 0;
        DB db = new DB();
        DataTable dt = new DataTable();
        SqlCommand cmd = new SqlCommand("SELECT UserID FROM Worker WHERE Username = @username", db.GetConnection());
        cmd.Parameters.Add("@username", SqlDbType.VarChar).Value = username;
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        adapter.Fill(dt);

        if (dt.Rows.Count > 0)
        {
            userID = (int)dt.Rows[0][0];
        }

        return userID;
    }
}

```

RequestID.Text overrides with requestID from previous form.

Also Method FillUserName() runs when the form initializes.

```

public void FillUsernameBox()
{
    DB db = new DB();
    DataTable dt = new DataTable();
    SqlCommand cmd = new SqlCommand("SELECT Username FROM Worker", db.GetConnection());
    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    adapter.Fill(dt);

    UsernameBox.DataSource = dt;
    UsernameBox.DisplayMember = "Username";
    UsernameBox.ValueMember = "Username";
}

1 reference
private void UpdateBtn_Click(object sender, EventArgs e)
{
    if (UsernameBox.SelectedIndex != -1)
    {
        string worker = UsernameBox.SelectedValue.ToString();
        int userID = GetUserIDByName(worker);

        string query = "UPDATE Request Set Worker = @userID WHERE RequestID = @requestID";

        DB db = new DB();
        SqlCommand cmd = new SqlCommand(query, db.GetConnection());
        cmd.Parameters.AddWithValue("@userID", userID);
        cmd.Parameters.AddWithValue("@requestID", RequestID.Text);

        db.OpenConnection();
        if (cmd.ExecuteNonQuery() == 1)
            MessageBox.Show("Request updated");
        else
            MessageBox.Show("Request haven't been updated");
        db.CloseConnection();
    }
}

```

Fill the username box fills the listbox with all the workers from Worker table. If statement in Update Button Click event checks if username listbox is not empty and Updates the Table.

Worker Main Menu

	RequestID	ServiceID	Requester	Request Statues	Urgent Statues	RequestQuantity
▶	2	2	Terence	New	<input checked="" type="checkbox"/>	25
	6	1	John	New	<input type="checkbox"/>	55
*					<input type="checkbox"/>	

Update

Close

In the worker main menu, all the request of the responding worker are show to him/her.

```
public partial class WorkerMainMenu : Form
{
    1 reference
    public WorkerMainMenu()
    {
        InitializeComponent();
        LoadData();
    }
    1 reference
    public int GetUserIDByName(string username)
    {
        int userID = 0;
        DB db = new DB();
        DataTable dt = new DataTable();
        SqlCommand cmd = new SqlCommand("SELECT UserID FROM Worker WHERE Username = @username", db.GetConnection());
        cmd.Parameters.Add("@username", SqlDbType.VarChar).Value = username;
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        adapter.Fill(dt);

        if (dt.Rows.Count > 0)
        {
            userID = (int)dt.Rows[0][0];
        }

        return userID;
    }
    1 reference
    private void LoadData()
    {

```



```

1 reference
private void LoadData()
{
    string username = LoginPage.Username;
    int workerID = GetUserIDByName(username);
    DB db = new DB();
    string query = $"SELECT * FROM Request where Worker = @workerID;";
    SqlCommand cmd = new SqlCommand(query, db.GetConnection());
    cmd.Parameters.AddWithValue("@workerID", workerID);
    SqlDataAdapter sda = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    DataGridSQL.DataSource = dt;
}

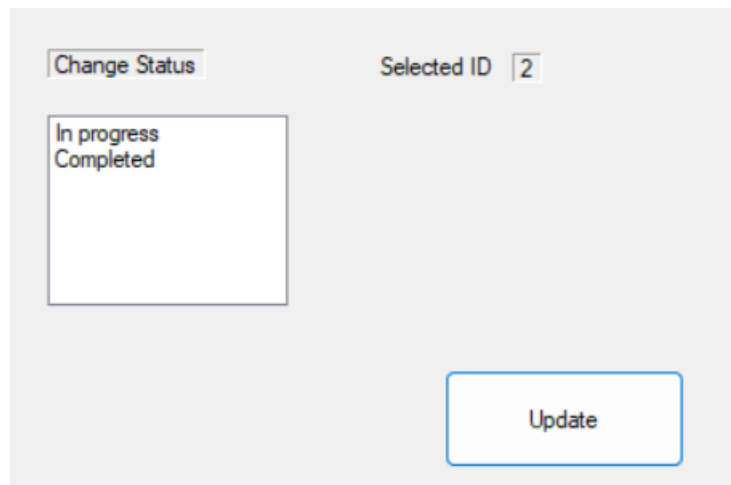
1 reference
private void CloseBtn_Click(object sender, EventArgs e)
{
    this.Close();
}

1 reference
private void UpdBtn_Click(object sender, EventArgs e)
{
    DataGridViewRow selectedRow = DataGridSQL.SelectedRows[0];
    int requestID = Convert.ToInt32(selectedRow.Cells["RequestID"].Value);
    WorkerStatusUpdate statusUpdate = new WorkerStatusUpdate(requestID);
    this.Hide();
    statusUpdate.ShowDialog();
    this.Close();
}

```

The codes for this form is similar to other Classes. There is an Update Button where worker can update Request Status of chosen request.

Worker Status Update



The form is titled "Worker Status Update". It contains a "Change Status" button at the top left. To its right is a "Selected ID" field with the value "2". Below the "Change Status" button is a status selection box with two options: "In progress" and "Completed". At the bottom right of the form is an "Update" button.

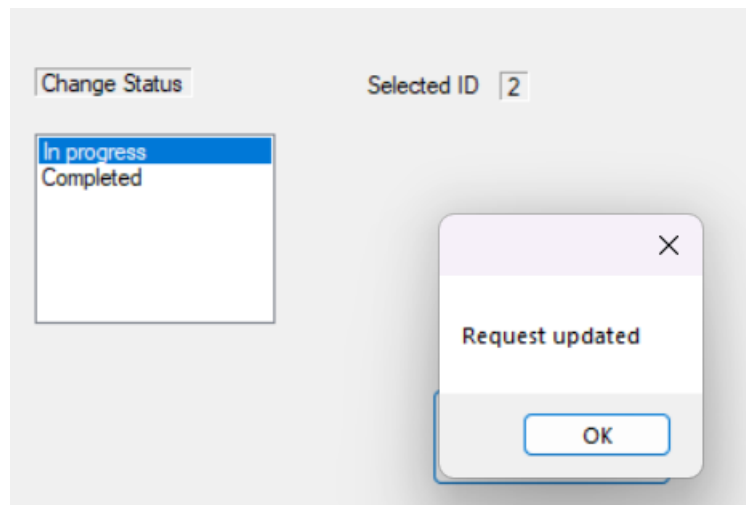
In this form worker can set the request statuses to “In progress” or “Completed” to indicate the worker progress on their assigned request.

```
private void button1_Click(object sender, EventArgs e)
{
    if (StatusBox.SelectedIndex != -1)
    {
        string status = "";
        if ((StatusBox.SelectedIndex == 0))
        {
            status = "In progress";
        }
        else if ((StatusBox.SelectedIndex == 1))
        {
            status = "Completed";
        }
        string query = "UPDATE Request Set RequestStatuses = @status WHERE RequestID = @requestID";
        DB db = new DB();
        SqlCommand cmd = new SqlCommand(query, db.GetConnection());
        cmd.Parameters.AddWithValue("@status", status);
        cmd.Parameters.AddWithValue("@requestID", RequestID.Text);

        db.OpenConnection();
        if (cmd.ExecuteNonQuery() == 1)
        {
            MessageBox.Show("Request updated");
        }
        else
        {
            MessageBox.Show("Request haven't been updated");
        }
        db.CloseConnection();
    }
    else
    {
        MessageBox.Show("Please select the status from the list box");
    }
}
```

RequestID	ServiceID	Requester	RequestStatuses	UrgentStatuses	RequestQuantity	Worker	PaymentStatus
2	2	Terence	New	1	25	3	Paid

The request table before update will show the request statues being “New”



RequestID	ServiceID	Requester	RequestStatues	UrgentStatues	RequestQuantity	Worker	PaymentStatus
2	2	Terence	In progress	1	25	3	Paid

After a successful update, the responding request will change its request statues to In Progress.

5. Test Plans and Cases

Test Case	Function Name	Test Objective	Expected Result	Actual Result	Remark
1	Login Form	To confirm the validation of user.	Message show the position of user for example "Manager" and receive a "Login Successfully" message.	All function and respond working successfully.	None
Manager					
2	Check work status	To check the work status by data grid view.	Able to check the status of the worker and the job and pop up "Assign Job" function when double click the column with handler function "Row header mouse double click".	Running smoothly with connection and synchronization of SQL server and Visual Studio. Display of data grid view and the handler function working.	None
3	Assign Job	To assign job to a specific worker.	Able to show the selection of worker name in the combo box. After selected and click the "Assign" button, the name of worker will update in the worker column in the data grid view in the Check work status form.	"Assign" button working and will update the worker's name into the data grid view.	None
4	Receive Payment	To calculate the bill to customer and distribute a receipt.	Able to calculate the total amount and print receipt to the customer.	"Calculate" and "Print Receipt" button working successfully.	None
Admin					
5	Main Admin Page	Is a main menu that can select task between three buttons "Register New User", "View Request Report" and "Monthly Income".	The function will run when the specific button has been selected.	All button function run successfully when it been selected.	None
6	Admin Register	Is a sign-up form with name, password, and user type	Able to update the user information when clicked the button "Register" and will update to the Account table in SQL server.	"Register" button working and able to update the information to SQL Account table.	None

7	Admin Request Report	Is a data grid view connected to SQL Account and Register table	Able to check the information of the new user.	Data grid view table show successfully.	None
8	Admin Monthly Income	Is a list box with usernames and can generate the personal monthly income report by select the name in the list box.	Able to select the name in the list box and the button "Generate" working.	Successfully generate monthly income report.	None
Worker					
9	Worker Main Menu	Is a data grid view connected to "request" and "Service" table in SQL	Can refers the data grid view table. The "Urgent" radio button will be able to update the status to manager check work status form.	"Urgent" radio button run successfully and will update to the manager check work status form.	None
10	Worker Status Update	Is built to update the work status to worker-to-worker main menu form.	Can select the work status from the list box and will update the status after clicked the "Update" button.	"Update" button function run successfully and able to update the worker status to the worker main menu form.	None
Customer					
11	Customer Main Menu	Is a main menu that connect between "New Request" form and "Customer Pending" form. User can choose the command by using the button given.	"Summit New Request" form will pop up when the button been clicked and "Customer Pending" form will pop up when the "Check Total Pending" button have been clicked.	"Summit New Request" and "Check Total Pending" buttons work successfully.	None
12	Customer New Request Form	Customer can release their order over this form.	Customer can select job task form the list box. After this, customer may be able to click the "Urgent" radio button if they are in urgent. The calculate button is for customer know the price before summit the order as a new request. Lastly,	The "Urgent" button function work successfully and will update the urgent status to manager "Check Work Status" form. "Calculate" button able to count the total	None

			customer may be able to click the “Summit New Request” button.	amount base on the price, quantity and urgent status.	
13	Customer Pending	Can check the pending work status and the information of customer from the data grid view table.	User can refer to the data grid view table when click the button “Check total Pending” from the “Customer Main Menu” form.	Able to review the table and the information was sync with SQL server request table.	None

6. Conclusion

In conclusion, the implementation of SQL connection functions as the foundation of our system, offering a safe and well-organized storage space for print job histories, system configurations, and student data, request detail, and system configurations. The usage of class, object and methods, have proven to be useful in our project due to its encouragement of reusability and modularity of concepts. OOP captures the essence of abstraction, enabling programmers to represent actual entities as objects. In academic settings, where clarity and conceptual comprehension are crucial, this abstraction makes it easier to depict complex systems in a way that is more intuitive and understandable. Object-oriented programming's benefits to the development of stable, expandable, and sustainable software products.

7. References

Thompson, B. (2023, October 21). *C# database connection: How to connect SQL server (example)*. Guru99. <https://www.guru99.com/c-sharp-access-database.html>

Chand, M. (2023, September 22). *DataTable in C#*. C# Corner. <https://www.c-sharpcorner.com/UploadFile/mahesh/datatable-in-C-Sharp/>

8. Workload Matrix

No.	Assigned Task & Bried Description	Assigned Member Name	Completion Status / Comment
1	Programming for the customer form, writing documentation and testing code.	Terence Lim Dao Liang	Work completed
2	Programming for the worker form, writing documentation and testing code.	Angelina Leanore	Work completed
3	Programming for the manager form, writing documentation and testing code	Tay Jun Long	Work completed
4	Programming for the admin form, writing documentation and debugging code.	Eraliev Suimonkul	Work completed