

EXAMPLE CODE
FOR
POINT OF SALES INFORMATION SYSTEM DEVELOPMENT PROJECT
REQUESTED BY
MR SALAD

COMPILED BY:

Group 10 - "Leaf Green IT Solutions"

COENRAAD HUMAN	28410629
HEINO NEL	26056984
PIETER BRAND	28633512
SAVANNAH FRITZE	29158710

DATE : 17TH SEPTEMBER 2018

TABLE OF CONTENTS

	Page
1.0 CODE EXAMPLES	
1.1 VALIDATION CODE	2-10
1.2 SQL STATEMENTS	10-10

1.0 VERIFYING INPUT

The screenshot displays the 'Mr Salad - Information System' application window. A modal dialog titled 'Add Client' is open, containing three sections: 'Personal Details', 'Contact Information', and 'Address Details'. In the 'Personal Details' section, the 'ID Number' field contains the value '0123456789'. A red error icon is visible to the right of this field, and a tooltip message 'Must be 13 digit ID.' is displayed. The 'Contact Information' section includes fields for 'Cell Number', 'Cell Number 2', and 'Email Address'. The 'Address Details' section includes fields for 'House Number', 'Street Name', 'Suburb', 'City Name' (with a dropdown menu showing 'Please select location.'), and 'Postal Code'. At the bottom of the dialog are 'Cancel' and 'Add' buttons.

Section	Field	Value	Validation
Personal Details	First Name	Verifying	
	Last Name	Input Test	
	ID Number	0123456789	Must be 13 digit ID.
Contact Information	Cell Number		
	Cell Number 2		
	Email Address		
Address Details	House Number		
	Street Name		
	Suburb		
	City Name	Please select location.	
	Postal Code		

Figure 1. Digit Length Verification

The screenshot displays a web application window titled "Mr Salad - Information System". The window has a menu bar with "File", "Edit", "Logout", "Windows", and "Help". In the center, there is a modal dialog box titled "Add Client". The dialog box contains several input fields organized into sections: "Personal Details" (First Name, Last Name, ID Number), "Contact Information" (Cell Number, Cell Number 2, Email Address), and "Address Details" (House Number, Street Name, Suburb, City Name, Postal Code). The "Email Address" field contains the text "!" and is highlighted with a red border and a red error icon. A tooltip message "Invalid Character!" points to the error icon. The "City Name" field is a dropdown menu with the text "Please select location." and a downward arrow. At the bottom of the dialog box are "Cancel" and "Add" buttons.

Section	Field	Value
Personal Details	First Name	Verifying
	Last Name	Input Test
	ID Number	0123456789012
Contact Information	Cell Number	0123456789
	Cell Number 2	0123456789
	Email Address	!
Address Details	House Number	
	Street Name	
	Suburb	
	City Name	Please select location.
	Postal Code	

Figure 2. Invalid Characters with regards to SQL injections

Mr Salad - Information System

File Edit Logout Windows Help

Add Client

Personal Details

First Name :

Last Name :

ID Number :

Contact Information

Cell Number :

Cell Number 2 :

Email Address :

Address Details

House Number :

Street Name :

Suburb :

City Name :

Postal Code :

Cancel Add

Must be a number!

Figure 3. Character Type Verification

The screenshot shows a web application window titled "Mr Salad - Information System". The main content area is a light gray background. A modal dialog box titled "Add Client" is centered on the screen. The dialog box contains three sections of form fields:

- Personal Details:**
 - First Name :
 - Last Name :
 - ID Number :
- Contact Information:**
 - Cell Number :
 - Cell Number 2 :
 - Email Address :
- Address Details:**
 - House Number :
 - Street Name :
 - Suburb :
 - City Name :
 - Postal Code :

At the bottom of the dialog box are two buttons: "Cancel" and "Add".

Figure 4. All fields have valid input

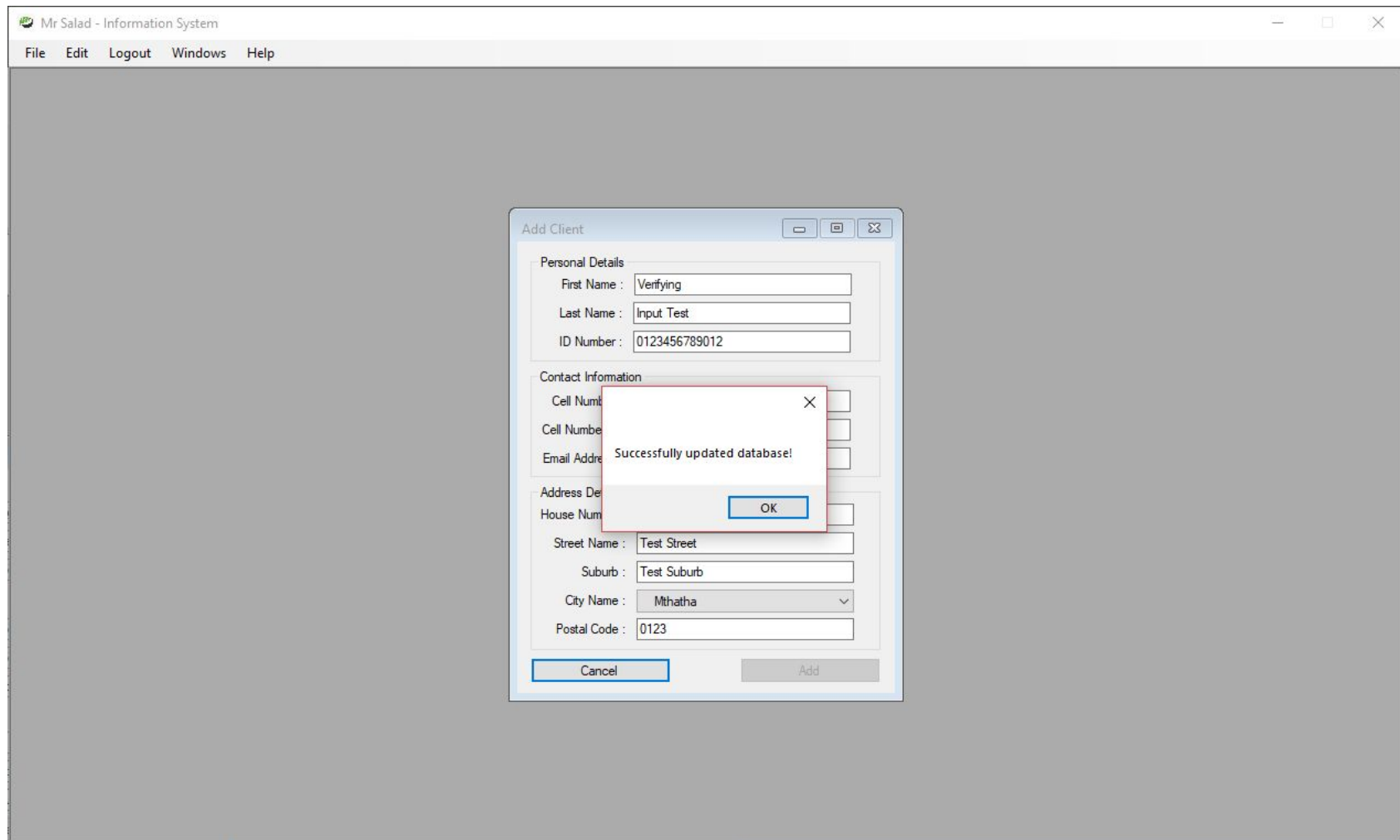


Figure 5. Database updated successfully

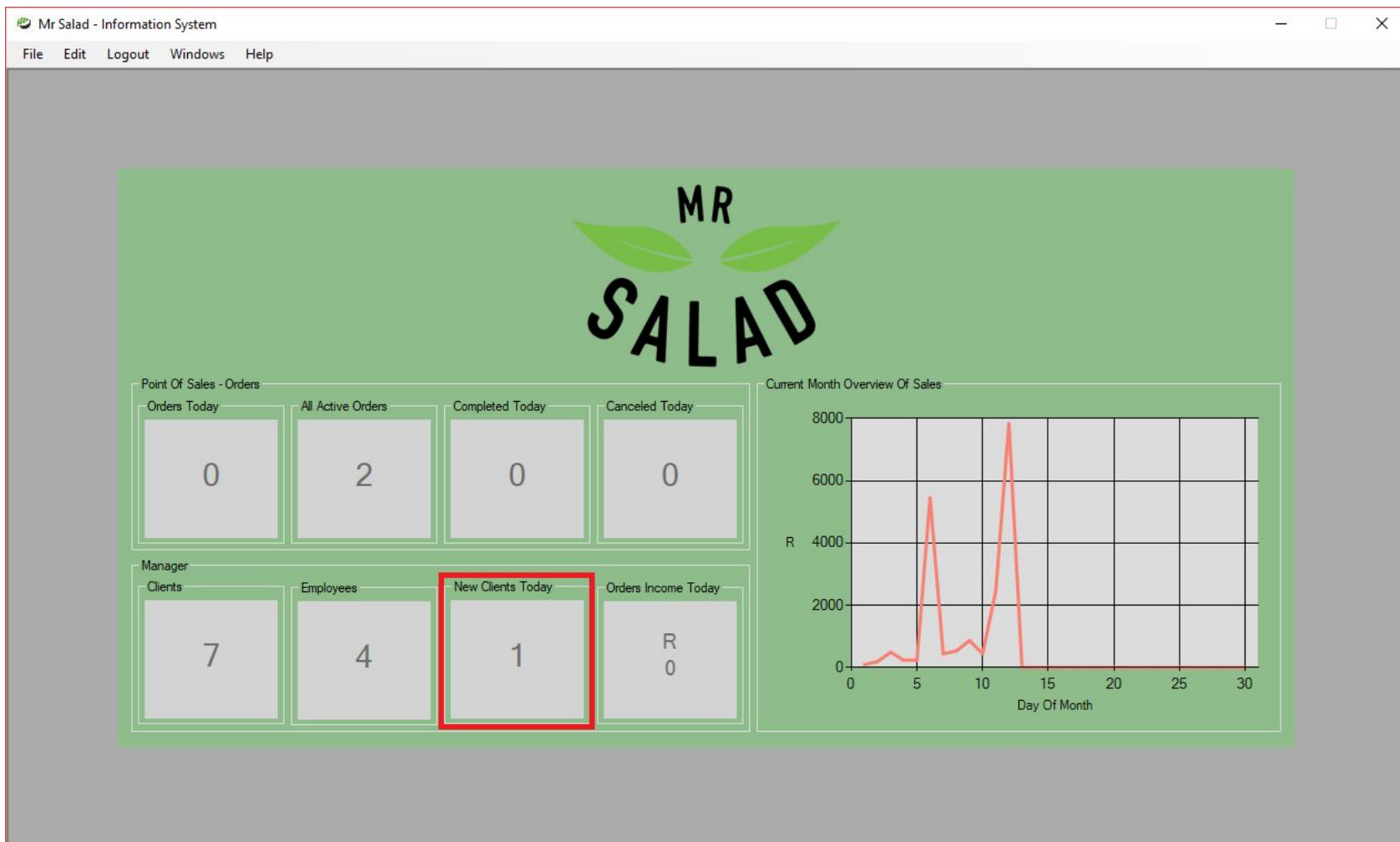


Figure 6. Updated dashboard

Database: Database- D:\Documents\itrw_225_project\TRW225_Information_System\T

File Home Create External Data Database Tools Fields Table Tell me what you want to do

View Paste Copy Cut Filter Ascending Descending Advanced Remove Sort Toggle Filter Refresh Save Delete More Find Replace Go To Select Text Formatting

All Access Objects

Search...

Tables

- CLIENT_ORDER
- CONTACT_DETAILS
- LOGIN
- ORDER_PRODUCTS
- PAYMENT_ORDER
- PERSON
- PERSON_TYPE
- PRODUCT
- SALES

Person_ID	Person_Name	Person_Surname	Person_Is_Removed	Person_Is_Employee	Person_Type	Person_Added
0123456789012	Verifying	Input Test	<input type="checkbox"/>	<input type="checkbox"/>	5	2018/09/16
2	Savannah	Fritze	<input type="checkbox"/>	<input checked="" type="checkbox"/>	4	2018/07/18
3	Pieter	Brand	<input type="checkbox"/>	<input checked="" type="checkbox"/>	4	2018/07/18
4	Heino	Nel	<input type="checkbox"/>	<input checked="" type="checkbox"/>	4	2018/07/18
4905469757456	Jan	Gubreg	<input type="checkbox"/>	<input type="checkbox"/>	5	2018/09/12
5406497957655	Pat	Sneiman	<input type="checkbox"/>	<input type="checkbox"/>	5	2018/09/12
5508165161651	Karmen	Hillton	<input type="checkbox"/>	<input type="checkbox"/>	5	2018/09/10
6408494675657	Karel	Rotkof	<input type="checkbox"/>	<input type="checkbox"/>	5	2018/09/07
8905747947653	Koos	Rosin	<input type="checkbox"/>	<input type="checkbox"/>	5	2018/09/08
9305105013084	Coenraad	Human	<input type="checkbox"/>	<input checked="" type="checkbox"/>	4	2018/07/18
9911079127912	Klaas	Raatgrief	<input type="checkbox"/>	<input type="checkbox"/>	5	2018/09/04
*			<input type="checkbox"/>	<input type="checkbox"/>	0	

Figure 7. Verifying updated input in database (PERSON table)

Database: Database- D:\Documents\itrw_225_project\TRW225_Information_System\TRW225_Information_System\bin\Debug\

File Home Create External Data Database Tools Fields Table Tell me what you want to do

View Paste Copy Cut Filter Ascending Descending Advanced Remove Sort Toggle Filter Refresh Save Delete More Find Replace Go To Select Text Formatting

All Access Objects

Search...

Tables

- CLIENT_ORDER
- CONTACT_DETAILS
- LOGIN
- ORDER_PRODUCTS
- PAYMENT_ORDER
- PERSON
- PERSON_TYPE
- PRODUCT
- SALES

HOUSE_NUMBER	STREET_NAME	POSTAL_CODE	Cell_Number_1	Cell_Number_2	SUBURB	CITY	Email_Address	Person_ID
21A	Test Street	123	123456789	123456789	Test Suburb	Mthatha	verifyinputtest@gmail.com	0123456789012
2		2	2	2	2	King William's Town	savannahtfritze@gmail.com	2
3		3	3	3	3	King William's Town	pbrand61@gmail.com	3
4		4	4	4	4	King William's Town	heino1369@gmail.com	4
16	Hendrik	4654	0649857754	0794954926	Hillbrow	Johannesburg	Jan@axcess.co.za	4905469757456
59	Jasel	4958	0796492424	0642492424	Rooiplaas	Bloemfontein	Pat@gmail.com	5406497957655
45	Lou	6456	0846219522	0794612462	Kampton	Koeberg	karmen@mweb.co.za	5508165161651
78	Klopper	5496	0794952426	0679579526	Lootweg	Kneisna	Karel@gmail.com	6408494675657
84	Kalma	5496	0794264926	0841967427	Braxton	Cape Town	koos@gmail.com	8905747947653
56	Lang	5467	0795654666	0812454635	Somewhere	King William's Town	coen.human@gmail.com	9305105013084
76	Kraamweg	5966	0697275245	0879462466	Rutfontein	Parys	Klaas@gmail.com	9911079127912
*								

Figure 8. Verifying updated input in database (CONTACT_DETAILS table)

```
private void ValidateComponent(TextBox textBox, CancelEventArgs e, ErrorProvider error)
{
    if (String.IsNullOrEmpty(textBox.Text))
    {
        e.Cancel = true;
        error.SetError(textBox, "Required field.");
    }
    else
    {
        if ( textBox.Text.Contains("'") || textBox.Text.Contains("\") || textBox.Text.Contains("|") || textBox.Text.Contains("-") ||
            textBox.Text.Contains("*") || textBox.Text.Contains("/") || textBox.Text.Contains("<>") || textBox.Text.Contains("<") ||
            textBox.Text.Contains(">") || textBox.Text.Contains(",") || textBox.Text.Contains("=") || textBox.Text.Contains("<=") ||
            textBox.Text.Contains(">=") || textBox.Text.Contains("~=") || textBox.Text.Contains("!=") || textBox.Text.Contains("^=") ||
            textBox.Text.Contains("(") || textBox.Text.Contains(")") )
        {
            e.Cancel = true;
            error.SetError(textBox, "Invalid Character!");
        }
        else
        {
            e.Cancel = false;
            error.SetError(textBox, null);
        }
    }
}
```

Figure 9. Code snippet of Validation on character entered

The method, ValidateComponent, used in Figure 9. is used to make sure when a special character is entered the program doesn't crash but rather throws an error asking to input a valid character.

```
private void ValidateEmail(TextBox textBox, CancelEventArgs e, ErrorProvider error)
{
    if (String.IsNullOrEmpty(textBox.Text))
    {
        e.Cancel = true;
        error.SetError(textBox, "Required field.");
    }
    else
    {
        if (checkEmail(textBox.Text))
        {
            e.Cancel = true;
            error.SetError(textBox, "Email already exists!");
        }
        else
        {
            if (textBox.Text.Contains("") || textBox.Text.Contains("\") || textBox.Text.Contains("|") || textBox.Text.Contains("-") ||
                textBox.Text.Contains("*") || textBox.Text.Contains("/") || textBox.Text.Contains("<>") || textBox.Text.Contains("<") ||
                textBox.Text.Contains(">") || textBox.Text.Contains(",") || textBox.Text.Contains("=") || textBox.Text.Contains("<=") ||
                textBox.Text.Contains(">=") || textBox.Text.Contains("~=") || textBox.Text.Contains("!=") || textBox.Text.Contains("^=") ||
                textBox.Text.Contains("(") || textBox.Text.Contains(")"))
            {
                e.Cancel = true;
                error.SetError(textBox, "Invalid Character!");
            }
            else
            {
                e.Cancel = false;
                error.SetError(textBox, null);
            }
        }
    }
}
```

Figure 10. Code Snippet of Validation on Email (Figure 2)

The method, ValidateEmail, used in Figure 10. is used to make sure when a user inputs their email and it isn't valid the program doesn't crash but rather throws an error asking the user to re-enter email. The method also checks if the email is already in use and whether the user has entered an email seeing as it's a required field.

Section 1

```
private void ValidateNumber(TextBox textBox, CancelEventArgs e, ErrorProvider error, string type)
{
    int length = 0;
    string msg = "";
    switch (type)
    {
        case "ID":
            length = 13;
            msg = "Must be 13 digit ID.";
            break;
        case "Cell":
            length = 10;
            msg = "Must be 10 digit cellphone number.";
            break;
        case "Postal":
            length = 4;
            msg = "Must be 4 digit postal code.";
            break;
        default:
            break;
    }
    if (String.IsNullOrEmpty(textBox.Text))
    {
        e.Cancel = true;
        error.SetError(textBox, "Required field.");
    }
    else
    {
        bool result = long.TryParse(textBox.Text, out long resultL);
```

Section 2

```
        if (result)
        {
            if (textBox.Text.Length != length)
            {
                e.Cancel = true;
                error.SetError(textBox, msg);
            }
            else
            {
                if (type == "ID")
                {
                    if (checkID(textBox.Text))
                    {
                        e.Cancel = true;
                        error.SetError(textBox, "ID already exists!");
                    }
                    else
                    {
                        e.Cancel = false;
                        error.SetError(textBox, null);
                    }
                }
                else
                {
                    e.Cancel = false;
                    error.SetError(textBox, null);
                }
            }
        }
        else
        {
            e.Cancel = true;
            error.SetError(textBox, "Must be a number!");
        }
    }
}
```

Figure 11. Code Snippet of Validation on input length

The method, ValidateNumber, is a method that checks the length of the characters entered by the users to make sure the string is valid. Also makes sure there is an input as the field is a required field.

2.0 SQL STATEMENTS

ADDING NEW CLIENT - CODE SNIPPET

default:

```
        buttonSave.Enabled = false;
        // this adds person
        using (OleDbConnection db = new
OleDbConnection(Properties.Settings.Default.DatabaseConnectionString))
        {
            string query = String.Format("INSERT INTO PERSON
(Person_ID,Person_Name,Person_Surname,Person_Is_Removed,Person_Is_Employee,Person_Type, Person_Added)
VALUES('{0}', '{1}', '{2}', False, False, 5, @1)",
            textBoxID.Text, textBoxFN.Text, textBoxLN.Text);
            db.Open();
            OleDbDataAdapter adapter = new OleDbDataAdapter("SELECT * FROM PERSON", db);
            OleDbCommand command = new OleDbCommand(query, db);
            command.Parameters.Add("@1", OleDbType.Date).Value = DateTime.Today;
            adapter.InsertCommand = command;
            adapter.InsertCommand.ExecuteNonQuery();
            db.Close();
        }
        // this add contact details
        using (OleDbConnection db = new
OleDbConnection(Properties.Settings.Default.DatabaseConnectionString))
        {
            string query = String.Format("INSERT INTO CONTACT_DETAILS (Person_ID, House_Number,
Street_Name, Postal_Code, Cell_Number_1, Cell_Number_2, Suburb, City, Email_Address) VALUES('{0}', '{1}', '{2}', {3},
{4}, {5}, '{6}', '{7}', '{8}')",
            textBoxID.Text, textBoxHN.Text, textBoxSN.Text, textBoxPC.Text, textBoxCN.Text,
textBoxCN2.Text, textBoxS.Text, comboBoxCN.SelectedItem.ToString(), textBoxEA.Text);
            db.Open();
```

```
OleDbDataAdapter adapter = new OleDbDataAdapter("SELECT * FROM PERSON", db);
OleDbCommand command = new OleDbCommand(query, db);
adapter.InsertCommand = command;
adapter.InsertCommand.ExecuteNonQuery();
db.Close();
}
MessageBox.Show("Successfully updated database!");
ClearTextBoxes();
comboBoxCN.SelectedIndex = 0;
buttonSave.Enabled = true;
break;
}
```

FORGOT PASSWORD AT LOGIN - CODE SNIPPET

```
try
{
    MailMessage mail = new MailMessage();
    SmtpClient SmtServer = new SmtpClient("smtp.gmail.com");

    mail.From = new MailAddress("leafgreenitsolutions.mrsalad@gmail.com");
    mail.To.Add(email);
    mail.Subject = "Mr Salad - Reset Password for " + email;
    BE_GeneratePassword pass = new BE_GeneratePassword();
    string password = pass.generate();
    mail.Body = "Your new password is: " + password;

    SmtServer.Port = 587;
    SmtServer.Credentials = new
System.Net.NetworkCredential("leafgreenitsolutions.mrsalad@gmail.com", "Google18!");
    SmtServer.EnableSsl = true;
    SmtServer.Send(mail);
}
```



```
        BE_DatabaseCommands dbCommands = new BE_DatabaseCommands();
        string query = String.Format("UPDATE LOGIN INNER JOIN CONTACT_DETAILS ON
LOGIN.Person_ID = CONTACT_DETAILS.Person_ID SET LOGIN.Password = '{0}' WHERE
CONTACT_DETAILS.Email_Address = '{1}'", dbCommands.hashPassword(password), email);
        dbCommands.updateDB(query, "LOGIN");

        return "New password sent: " + email;
    }
    catch (Exception ex)
    {
        BE_LogSystem log = new BE_LogSystem(ex);
        log.saveError();
        return "Reset password was not sent: " + email;
    }
}
```

USER MAINTENANCE UPDATE - CODE SNIPPET

```
using (OleDbConnection database = new OleDbConnection(Properties.Settings.Default.DatabaseConnectionString))
{
    BE_DatabaseCommands dbCommands = new BE_DatabaseCommands();
    database.Open();
    OleDbDataAdapter adapter = new OleDbDataAdapter("SELECT * FROM LOGIN", database);
    OleDbCommand command = new OleDbCommand(String.Format("UPDATE LOGIN SET
[A_CLIENT_MAINTENANCE] = @0, [A_EMPLOYEE_MAINTENANCE] = @1, [A_POINTS_OF_SALE] = @2,
[A_REPORTS] = @3, [A_USER_MAINTENANCE] = @4, [A_SETTINGS] = @5, [PASSWORD] = '{0}' WHERE
[EMPLOYEE_ID] = {1}", dbCommands.hashPassword(password), employeeID), database);
    command.Parameters.Add("@0", OleDbType.Boolean).Value = list[0];
    command.Parameters.Add("@1", OleDbType.Boolean).Value = list[1];
    command.Parameters.Add("@2", OleDbType.Boolean).Value = list[2];
    command.Parameters.Add("@3", OleDbType.Boolean).Value = list[3];
    command.Parameters.Add("@4", OleDbType.Boolean).Value = list[4];
}
```

```
        command.Parameters.Add("@5", OleDbType.Boolean).Value = list[5];
        adapter.InsertCommand = command;
        adapter.InsertCommand.ExecuteNonQuery();
        database.Close();
    }
    return "Updated permissions and password!";
}
catch (Exception ex)
{
    BE_LogSystem log = new BE_LogSystem(ex);
    log.saveError();
    return "Failed updating permission and password!";
}
}
```