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

1.0 VERIFYING INPUT

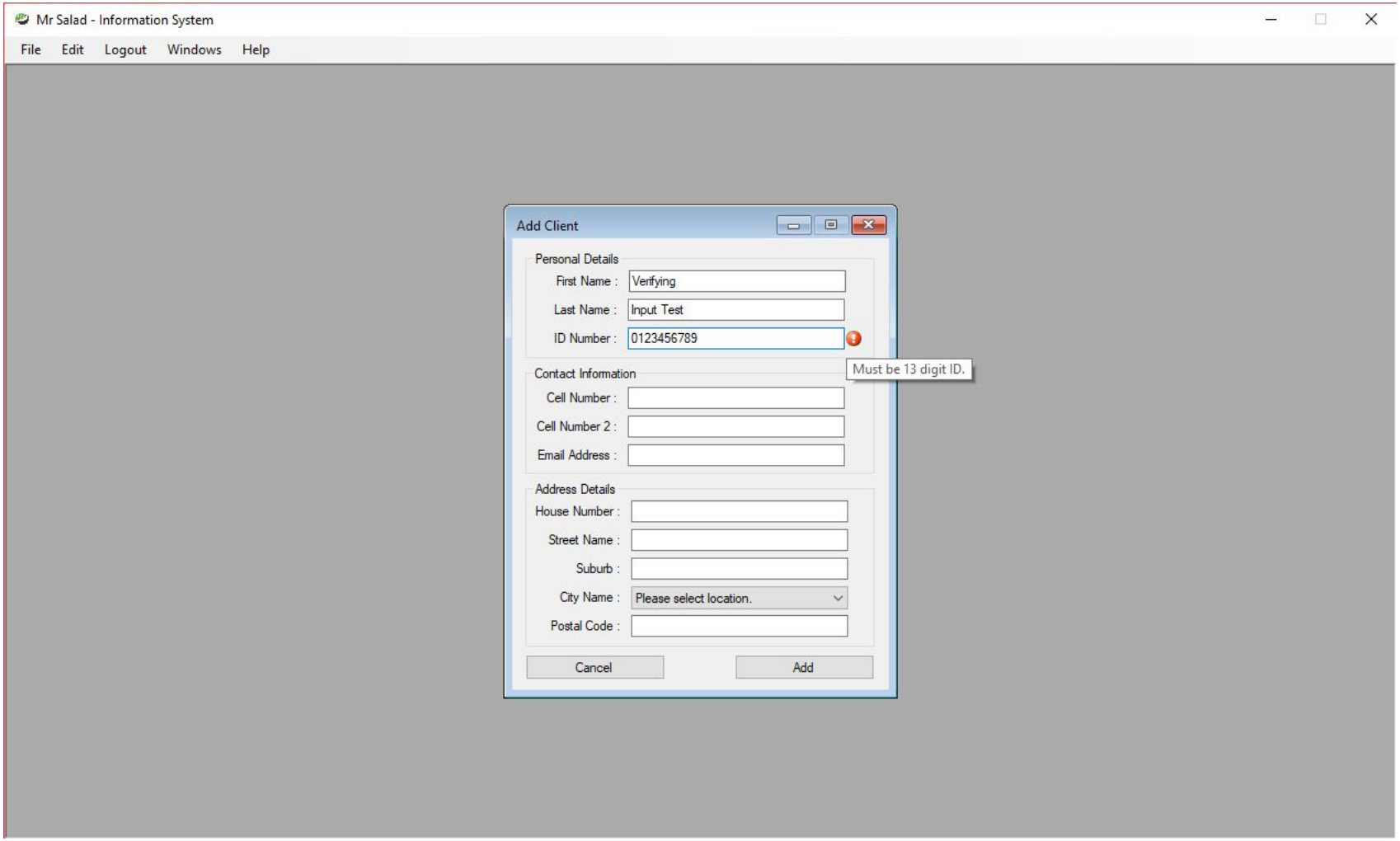

Figure 1. Digit Length Verification

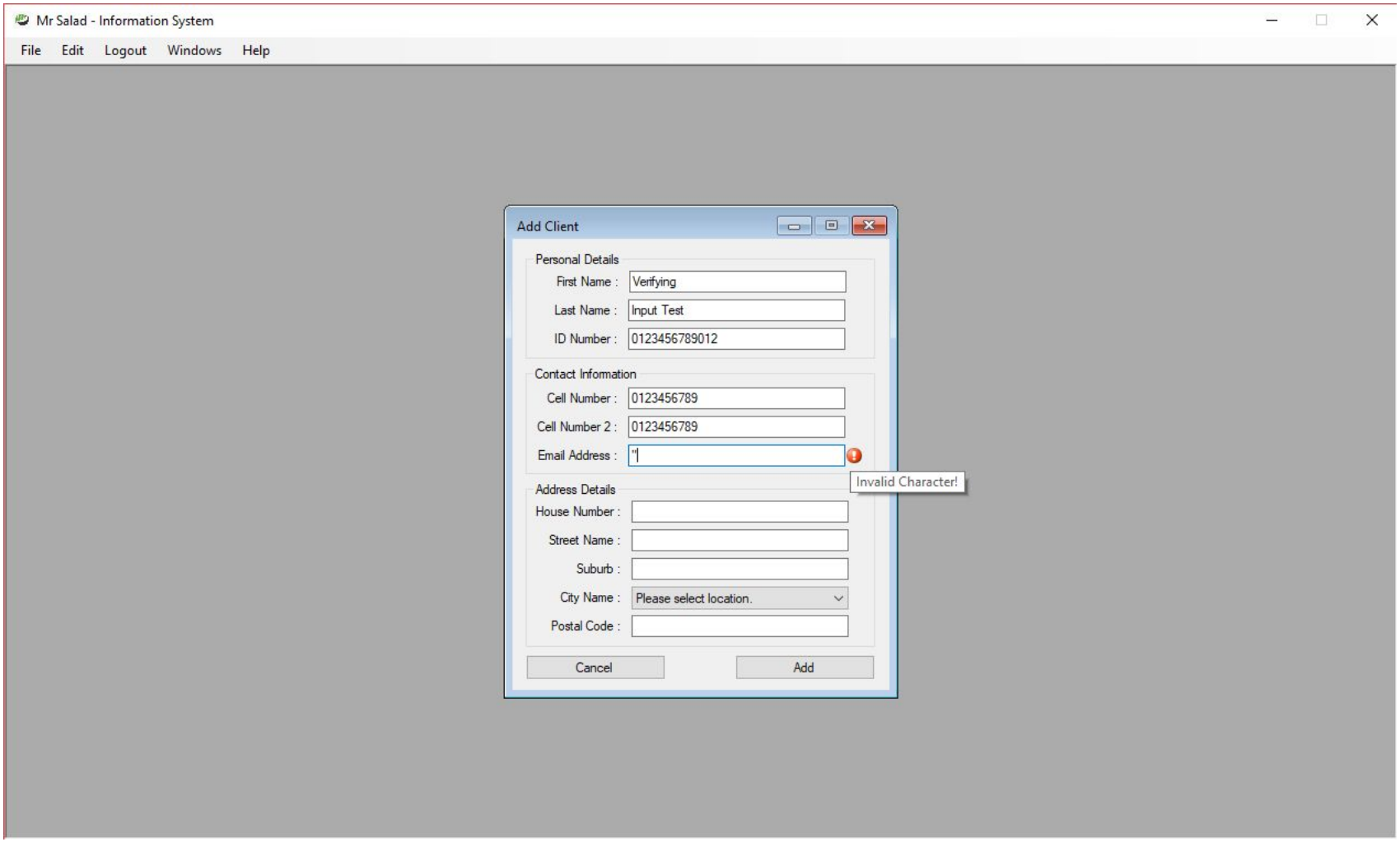

Figure 2. Invalid Characters with regards to SQL injections

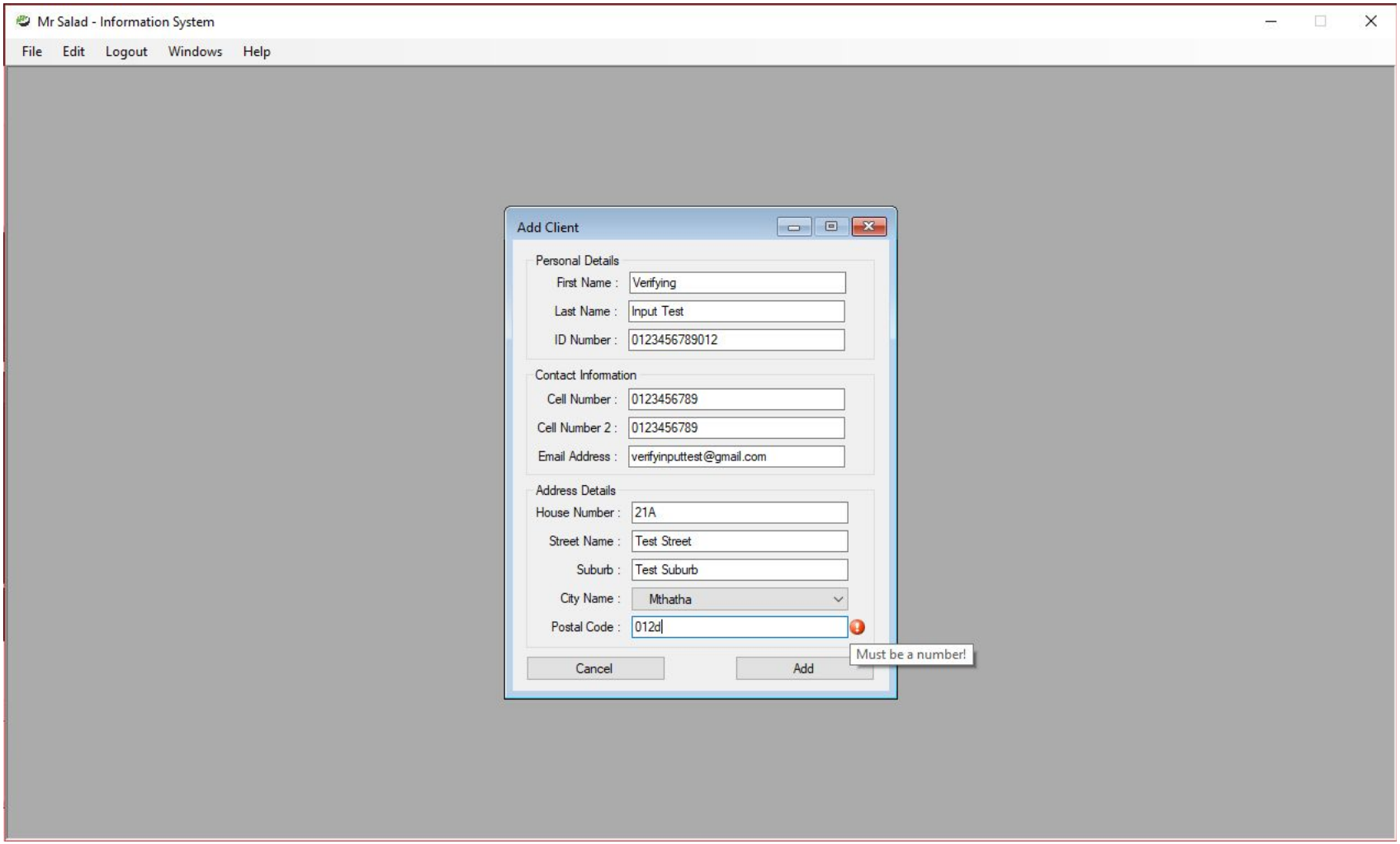

Figure 3. Character Type Verification

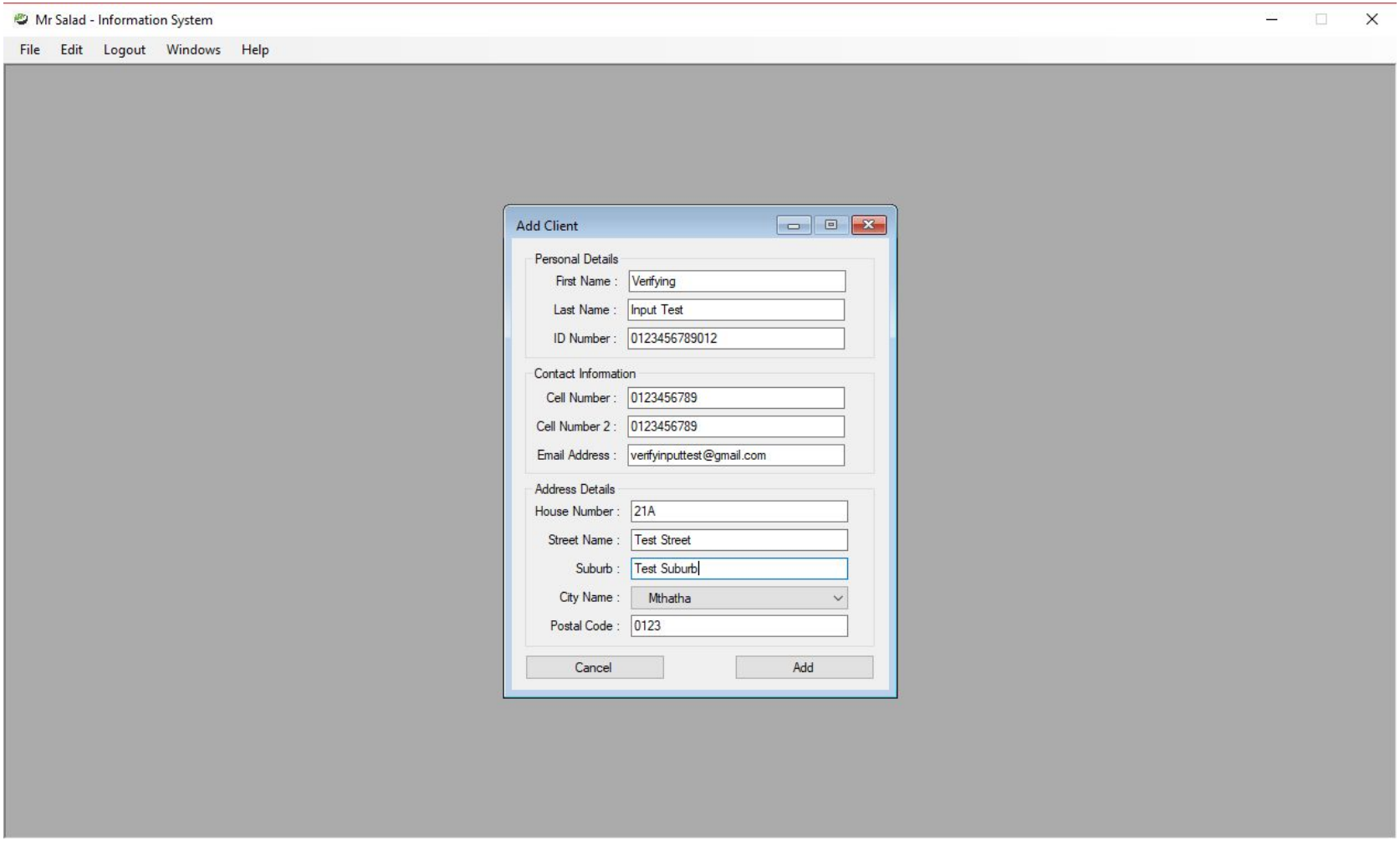

Figure 4. All fields have valid input

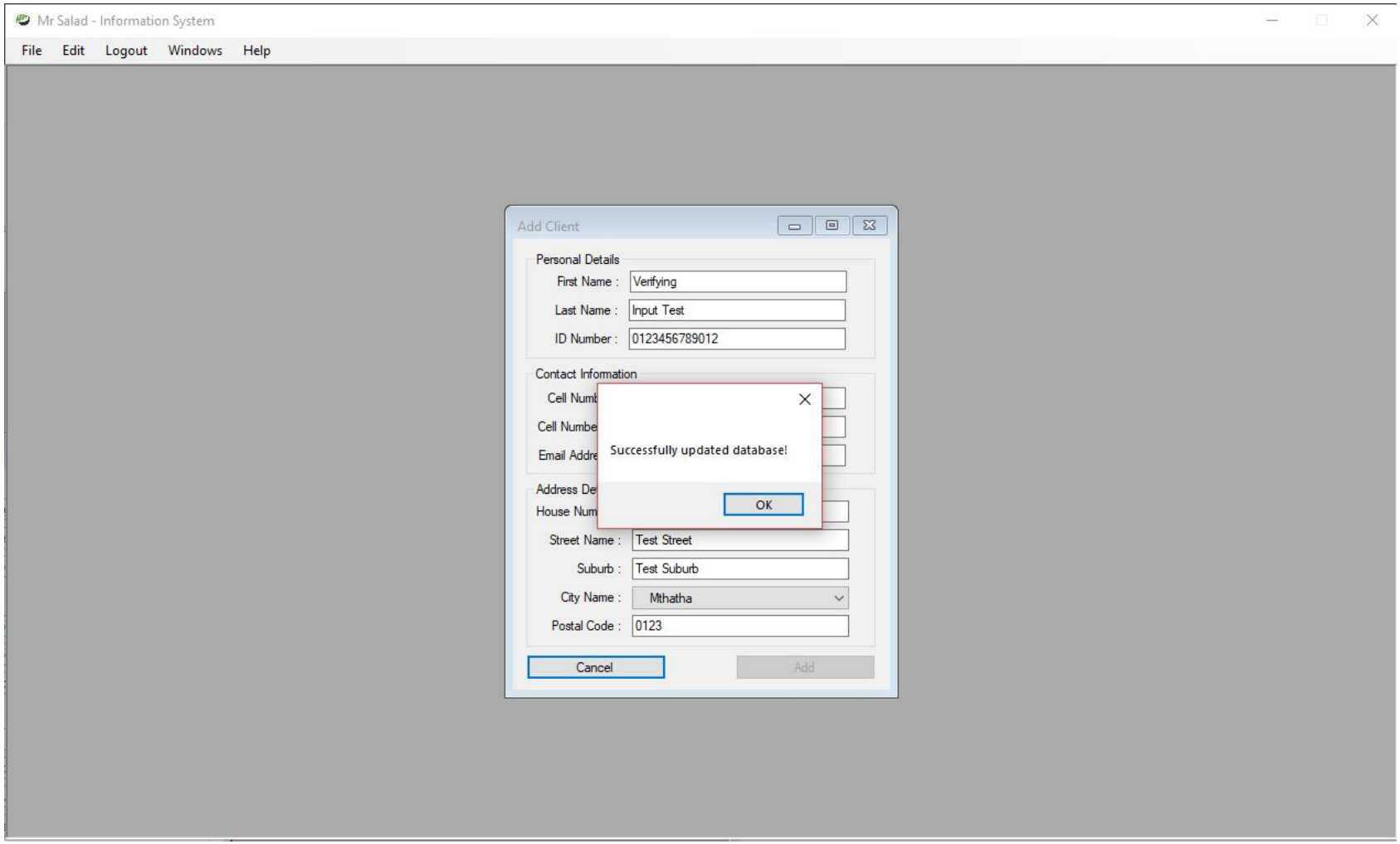

Figure 5. Database updated successfully

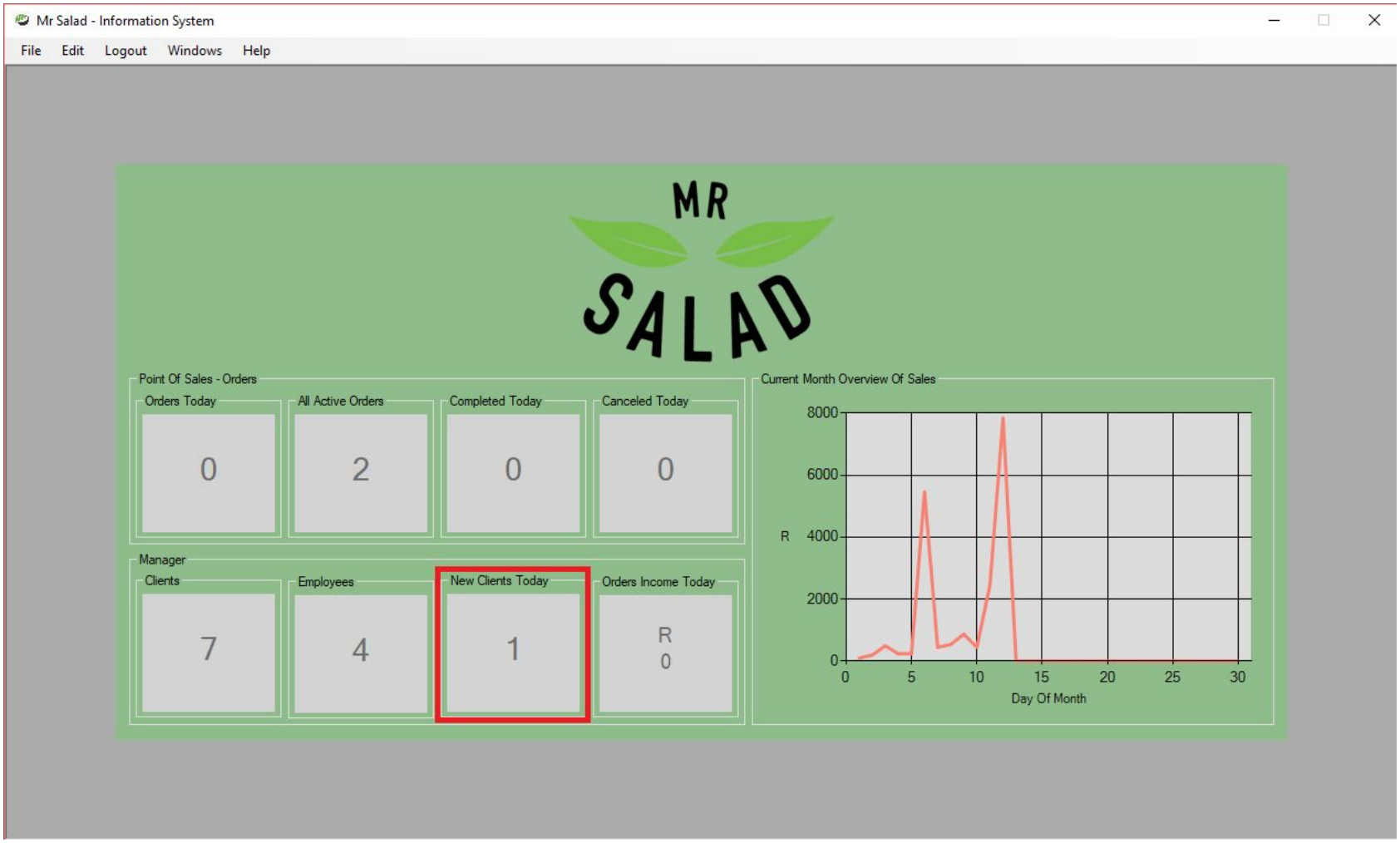

Figure 6. Updated dashboard

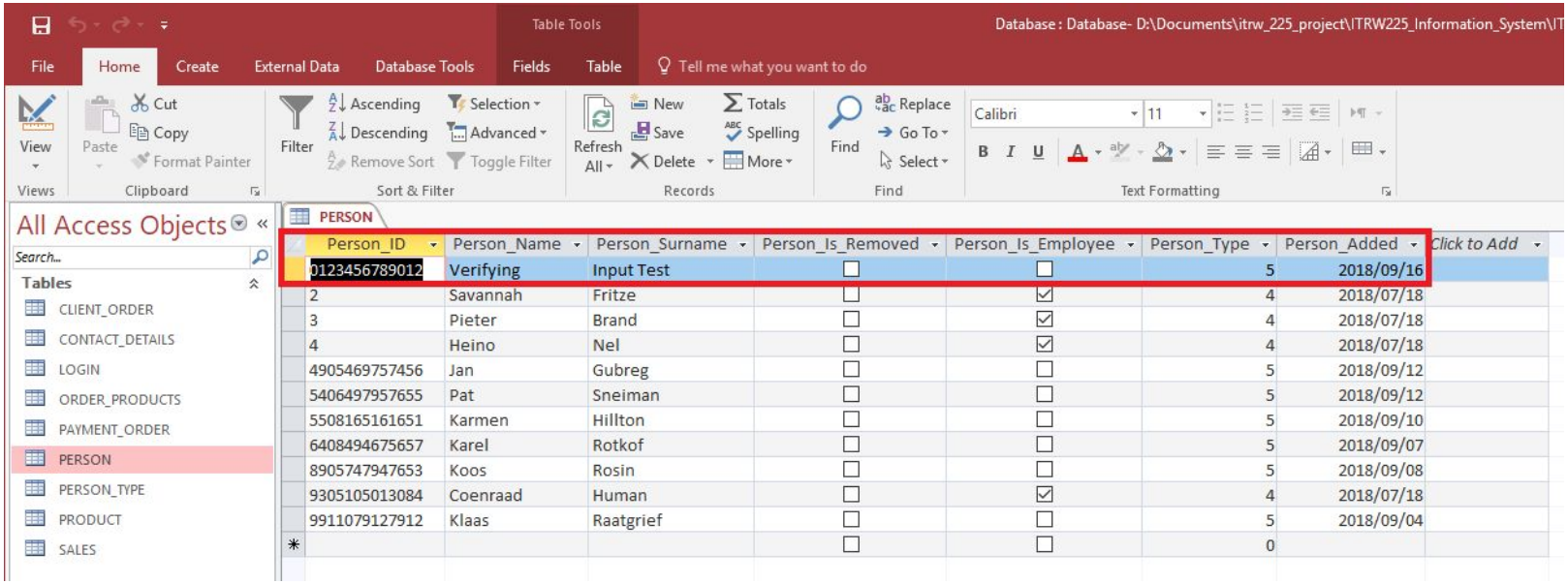

Figure 7. Verifying updated input in database (PERSON table)

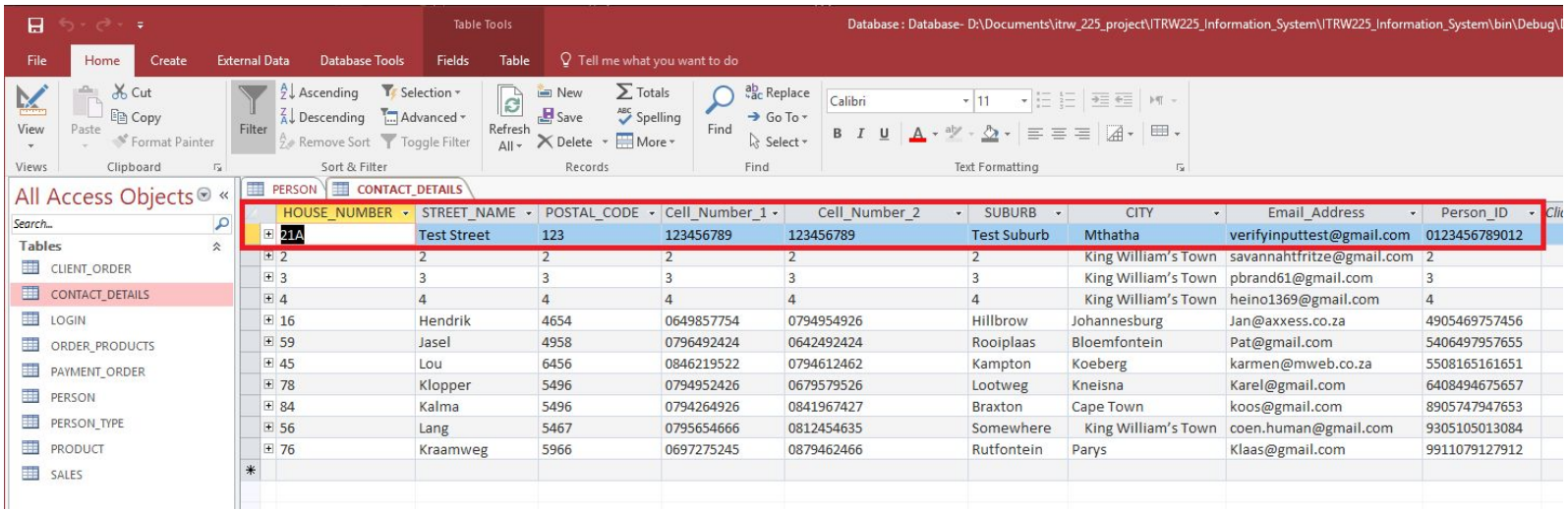

Figure 8. Verifying updated input in database (CONTACT_DETAILS table)

```
private void ValidateComponent(TextBox textBox, CancelEventArgs e, ErrorProvider error)
{
    if (String.IsNullOrWhiteSpace(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.IsNullOrWhiteSpace(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.IsNullOrWhiteSpace(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 SmtpServer = 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;

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