E-COMMERCE MANAGEMENT SYSTEM IMPLEMENTATION

SWE5201 – ADVANCED PROGRAMMING

REPORT DONE BY:
LES PAUL RANALAN
(2230727)

TO BE SUBMITTED TO:
MARIA ABEDI

DATE OF SUBMISSION:
APRIL 26, 2024

GITHUB REPOSITORY

https://github.com/shadowisf/CircuitCentral

TABLE OF CONTENTS

TABLE OF FIGURES

INTRODUCTION

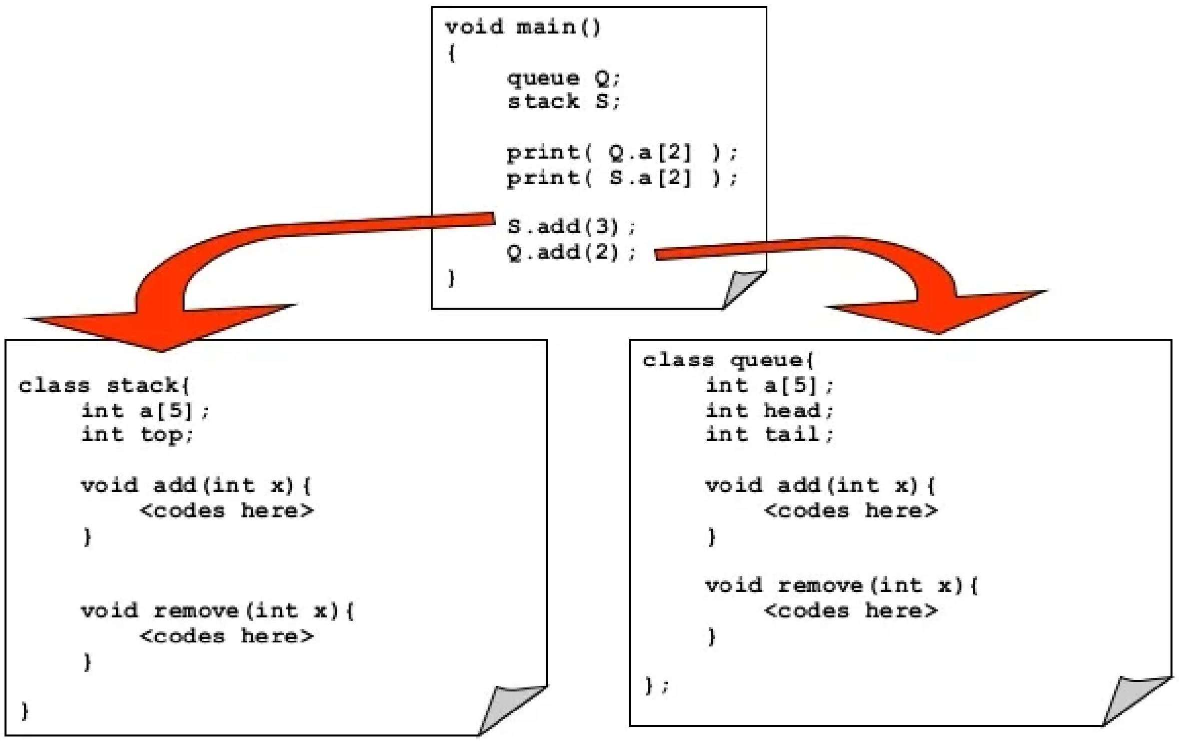
This report entails into building a C# computer program for managing an e-commerce management system. The C# language along with Windows Forms will be used to create this program. The main goal here is to show how versatile and robust C# can be when it comes to building this kind of program

The name of the program is **CircuitCentral** which is an e-commerce application where it sells electronic products.

Along with that, challenges that have arised during coding will be discussed thoroughly, with provided solutions.

OBJECT-ORIENTED PROGRAMMING CONCEPTS

Object-oriented programming (**OOP**) is a way of organizing code into reusable segments called “objects” **(Doherty, 2020)**. This helps in writing code that is more easier to understand and modify in the future. Such concepts are encapsulation, inheritance, and polymorphism. OOP will be used to create the C# system.



```
void main()
{
     queue Q;
     stack S;

     print( Q.a[2] );
     print( S.a[2] );

     S.add(3);
     Q.add(2);
}
```

```
class stack{
     int a[5];
     int top;

     void add(int x){
          <codes here>
     }


     void remove(int x){
          <codes here>
     }
}
```

```
class queue{
     int a[5];
     int head;
     int tail;

     void add(int x){
          <codes here>
     }

     void remove(int x){
          <codes here>
     }

};
```

Figure 1: Object-oriented code example

METHODOLOGY

USER INTERFACE – WINFORMS

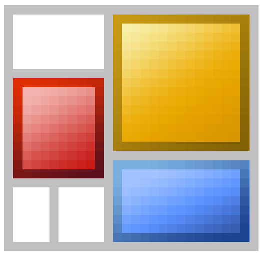


Figure 2: WinForms

Windows Forms (WinForms) is a graphical user interface framework for developing desktop applications on the Windows operating system **(adegeo, 2023)**, allowing for the creation of interactive applications with an easy-to-use design.

This project is based on a Windows Forms Application (**WFA**) which is a standard template that is present in Visual Studio 2022 (not to be confused with Visual Studio Code (VSCODE).

In a WFA, there are elements that are pre-defined for the user to just drag-and-drop onto the workspace. The project utilizes these elements in order to function properly, from clickable buttons to DataGridView tables.

Below are screenshots of the UI design of the project.

(next page)

LOGIN, REGISTER

The very first form that users will see is the Login and Register forms, which are used to verify if user is customer or an admin.

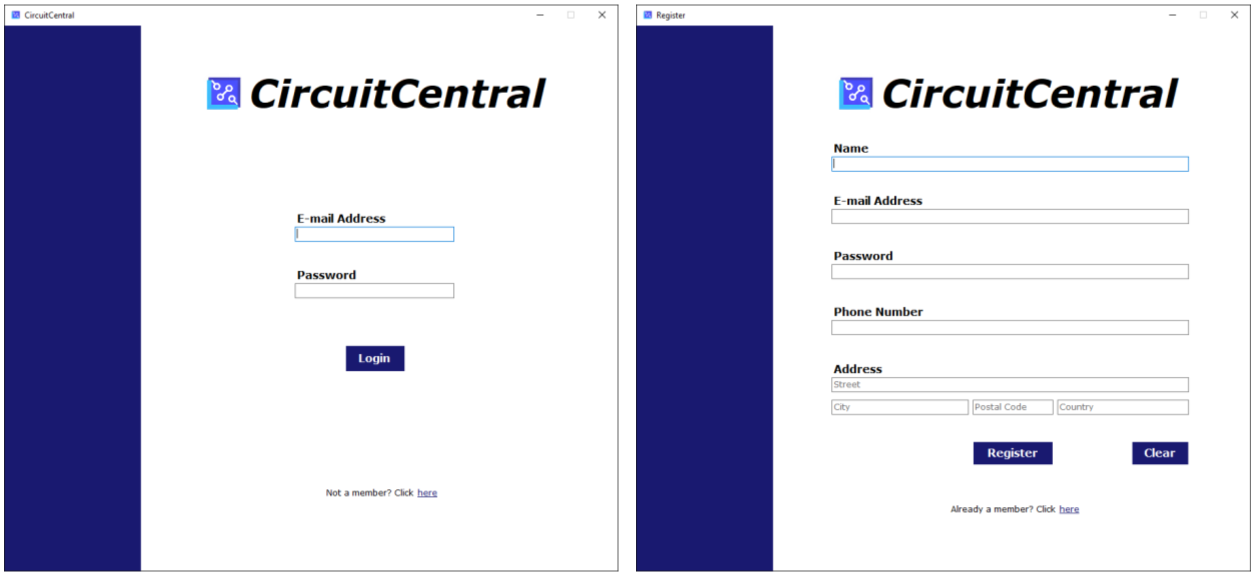


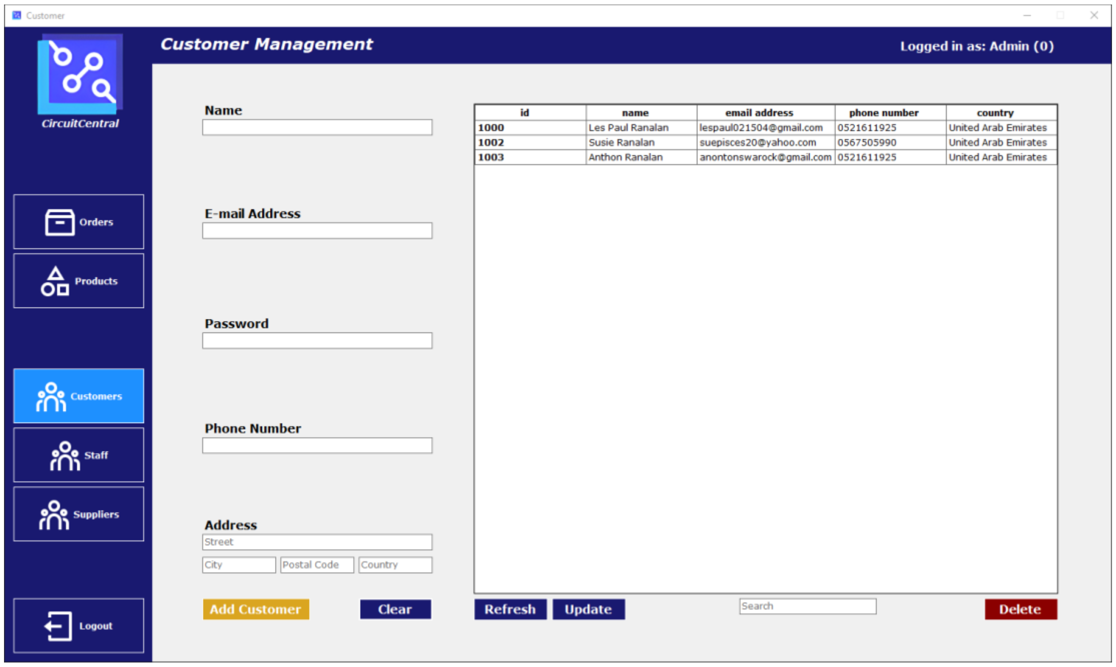
Figure 3: Login (left) and Register (right) forms

These forms feature very basic text boxes as input fields for the user. Along with that, buttons to execute designated SQL queries.

If ever customer is not registered, users can click the link label found at the very bottom of the form to redirect to the Register form and enter necessary credentials for the user's account. Afterwards, users can redirect to the Login form again, via link label at the bottom, to login to the system.

CUSTOMER, STAFF, SUPPLIER

When user is logged in, it will redirect them to their respective forms based on user type. In this case, we are logged in as admin, it redirects us directly to the Customer Management form, with full priviledges to ALL forms.



id	name	email address	phone number	country
1000	Les Paul Ranalan	lespaul021504@gmail.com	0521611925	United Arab Emirates
1002	Susie Ranalan	suepisces20@yahoo.com	0567505990	United Arab Emirates
1003	Anthon Ranalan	anontonswarock@gmail.com	0521611925	United Arab Emirates

Figure 4: Customer Management

The Customer Management form features basic text boxes that acts as fields and various buttons for the user to interact with the database. Buttons and fields such as Add, Clear, Refresh, Update, Search, and Delete are used to perform CRUD operations within Customer table.

This form is similar to Staff and Supplier forms, so the design is repeated with minor changes to input fields and buttons. Similarly to Customer form, Staff and Supplier forms feature CRUD operations to interact with their respective tables within the database.

PRODUCT

When it comes to Product Management form, customers and admin have different views on this form. Note that when user's user type is customer and have a successfu login, it will redirect them directly to the Product Management form.

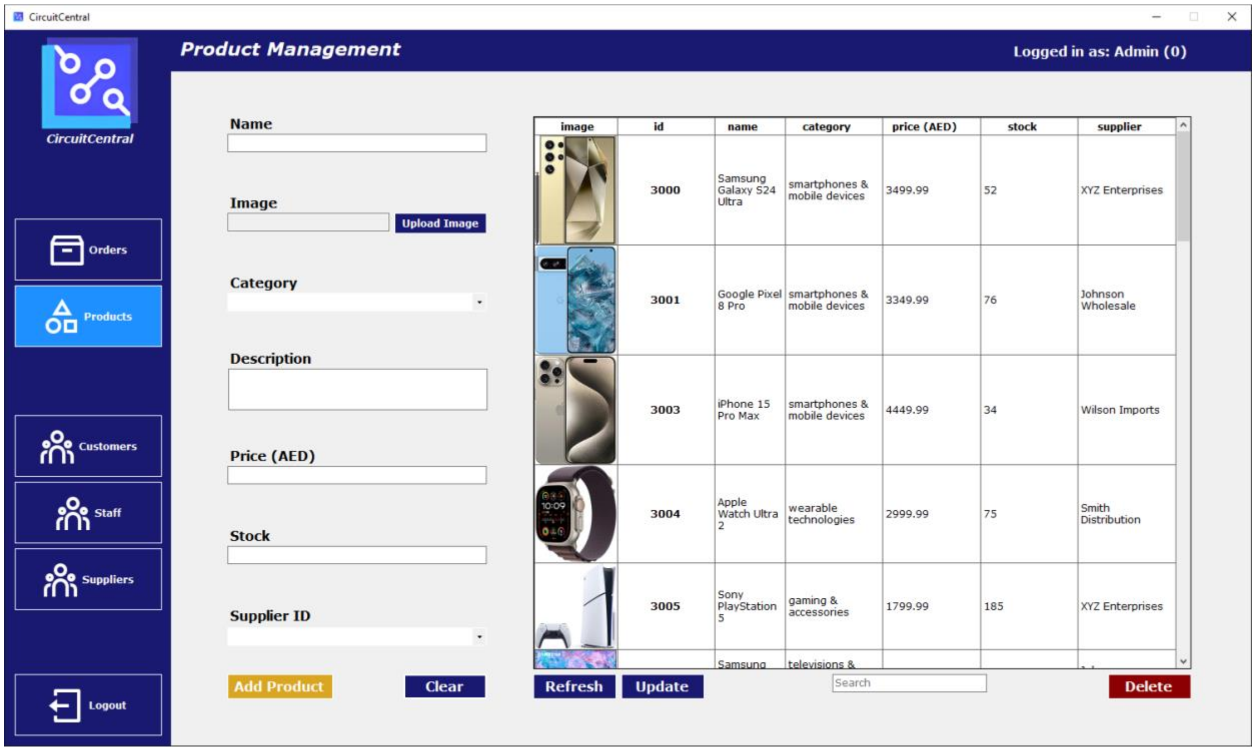
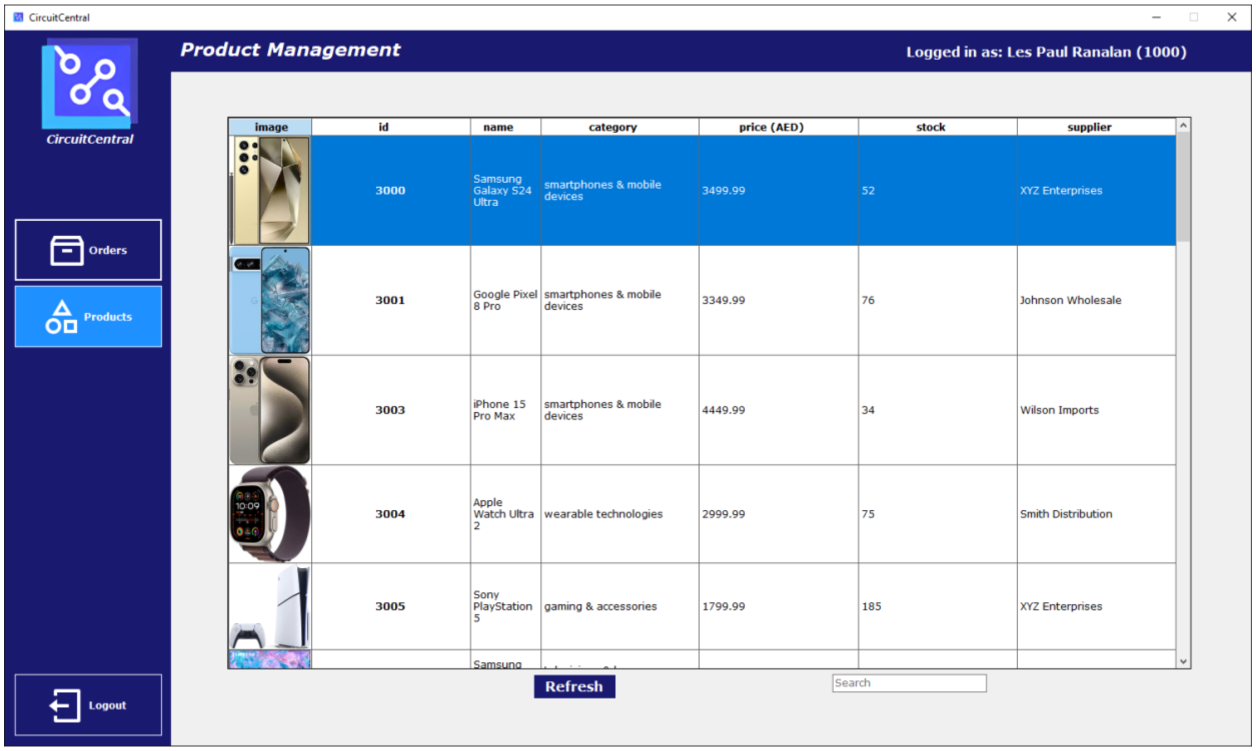


image	id	name	category	price (AED)	stock	supplier
	3000	Samsung Galaxy S24 Ultra	smartphones & mobile devices	3499.99	52	XYZ Enterprises
	3001	Google Pixel 8 Pro	smartphones & mobile devices	3349.99	76	Johnson Wholesale
	3003	iPhone 15 Pro Max	smartphones & mobile devices	4449.99	34	Wilson Imports
	3004	Apple Watch Ultra 2	wearable technologies	2999.99	75	Smith Distribution
	3005	Sony PlayStation 5	gaming & accessories	1799.99	185	XYZ Enterprises
		Samsung	televisions &			

Figure 5: Product Management (admin view)

As admin, we have full priviledges on Product Management form with text boxes as input fields along with buttons for CRUD operations. Additionally, this form's table features an image. This works by uploading the image to the system, although it does not actually upload the picture itself, but rather the directory to the specified image. Through that, with the use of codes and the image directory, it displays the image onto the table.

(next page)



Product Management

Logged in as: Les Paul Ranalan (1000)

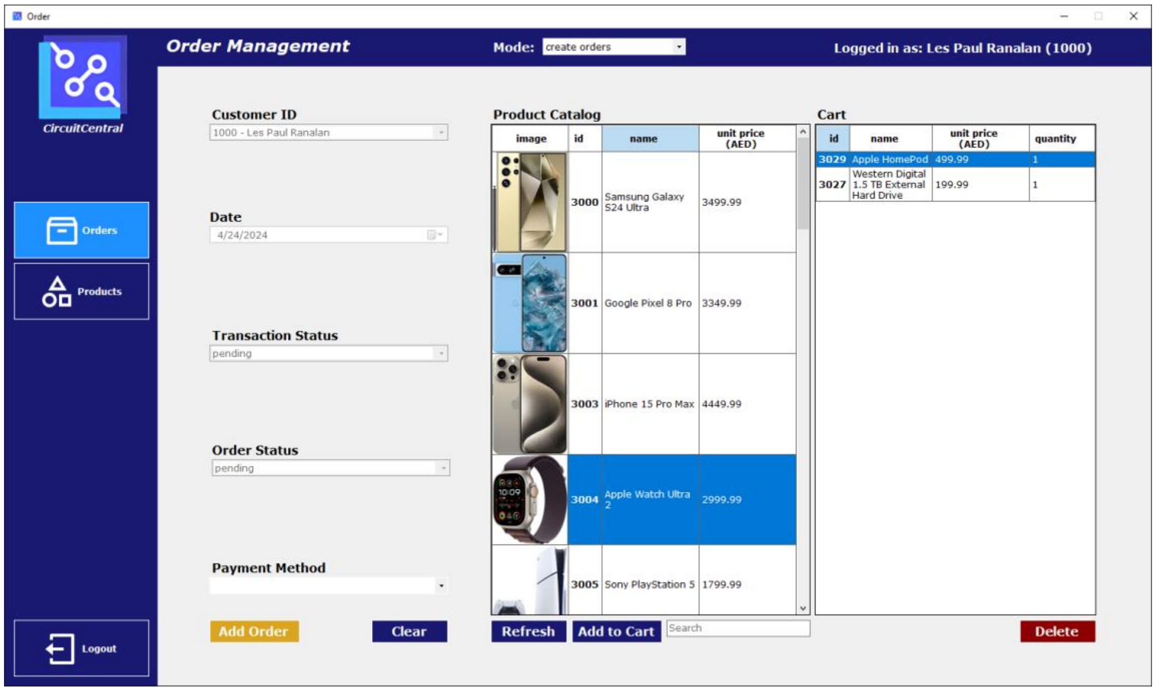
image	id	name	category	price (AED)	stock	supplier
	3000	Samsung Galaxy S24 Ultra	smartphones & mobile devices	3499.99	52	XYZ Enterprises
	3001	Google Pixel 8 Pro	smartphones & mobile devices	3349.99	76	Johnson Wholesale
	3003	iPhone 15 Pro Max	smartphones & mobile devices	4449.99	34	Wilson Imports
	3004	Apple Watch Ultra 2	wearable technologies	2999.99	75	Smith Distribution
	3005	Sony PlayStation 5	gaming & accessories	1799.99	185	XYZ Enterprises
		Samsung				

Figure 6: Product Management form (customer view)

As customer, we do not have full priviledges on Product Management form as we can only view the product catalog with no CRUD operations. This is only used when customer is browsing the product table without necessarily creating orders yet. This works by detection of user type and then making specific elements not visible when user is customer.

ORDER

Simiarly to Order Management form, there are 2 views based on user type. However, these views are only present when the user is on a specific mode within this form.



Order Management

Mode: create orders

Logged in as: Les Paul Ranalan (1000)

Customer ID: 1000 - Les Paul Ranalan

Date: 4/24/2024

Transaction Status: pending

Order Status: pending

Payment Method:

Product Catalog

image	id	name	unit price (AED)
	3000	Samsung Galaxy S24 Ultra	3499.99
	3001	Google Pixel 8 Pro	3349.99
	3003	iPhone 15 Pro Max	4449.99
	3004	Apple Watch Ultra 2	2999.99
	3005	Sony PlayStation 5	1799.99

Cart

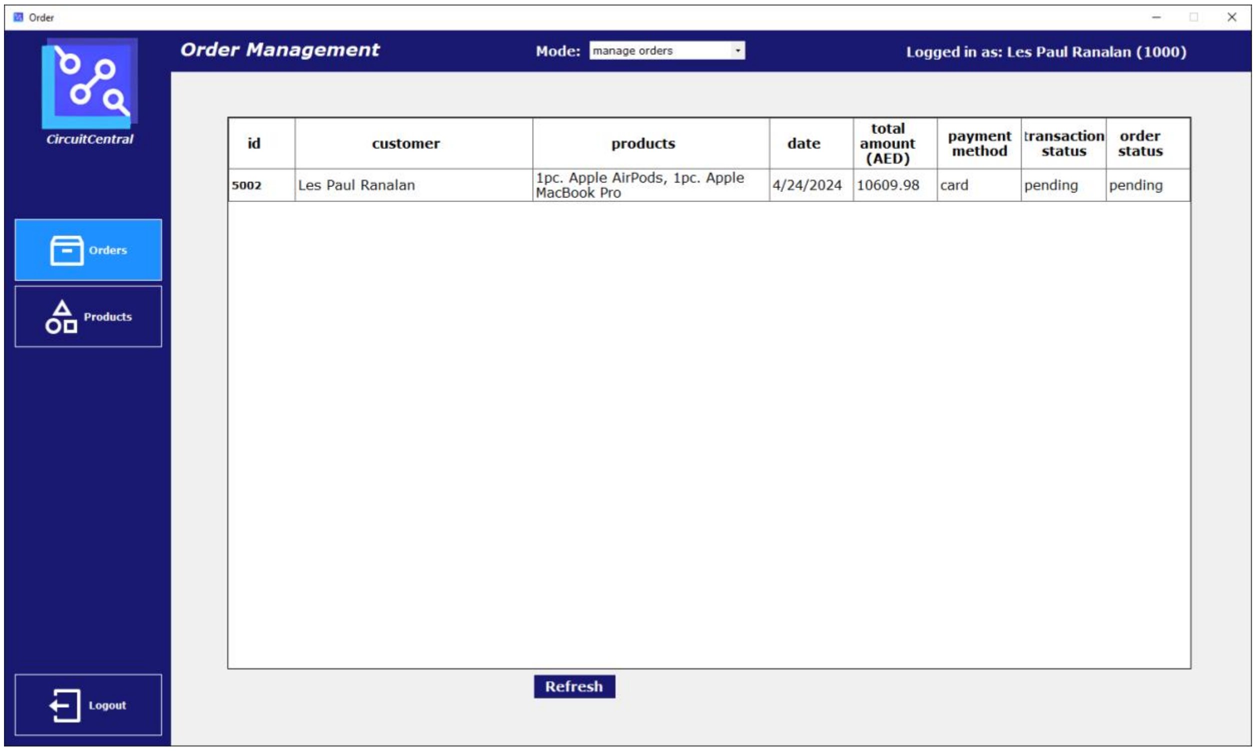
id	name	unit price (AED)	quantity
3029	Apple HomePod	499.99	1
3027	Western Digital 1.5 TB External Hard Drive	199.99	1

Figure 7: Order Management form (create orders mode)

At the very top of the form, there is a mode selector. Currently, we are logged in as customer and the mode is set to "create orders". Note that both admin and customer have the same exact view if mode is set to "create orders". However, when logged in as a customer, the Customer ID selector is locked to the user's exact ID and name while in admin view, it is not.

This form also features many tables which are the Product Catalog, Cart, and Order table. Along with text fields for user input and buttons for CRUD operations. The basic workings of this form goes like this: add products from the catalog to the cart, insert customer details and add order.

Once order is created, customer should be able to track their orders; this is done by changing the mode selector to “manage orders”. This is where customer and admin have different views on this specific mode.



Mode: manage orders

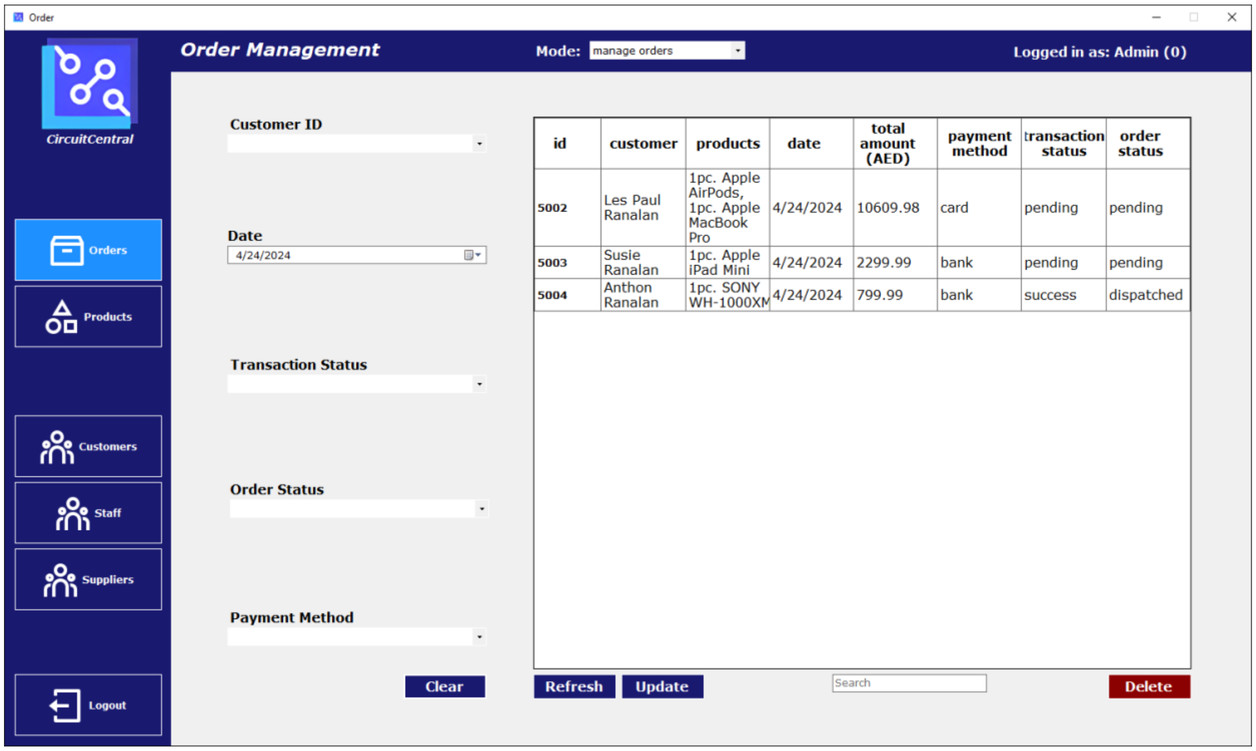
Logged in as: Les Paul Ranalan (1000)

id	customer	products	date	total amount (AED)	payment method	transaction status	order status
5002	Les Paul Ranalan	1pc. Apple AirPods, 1pc. Apple MacBook Pro	4/24/2024	10609.98	card	pending	pending

Figure 8: Order Management (manage orders mode, customer view)

As a customer with mode selector as “manage orders”, we are able to see and track our order. However, it does not feature any buttons and text fields for CRUD operations. Additiioanlly, as a customer, we are only able to track OUR OWN orders and not other customer’s orders.

(next page)



Order

Order Management

Mode: manage orders

Logged in as: Admin (0)

CircuitCentral

Customer ID:

Date: 4/24/2024

Transaction Status:

Order Status:

Payment Method:

id	customer	products	date	total amount (AED)	payment method	transaction status	order status
5002	Les Paul Ranalan	1pc. Apple AirPods, 1pc. Apple MacBook Pro	4/24/2024	10609.98	card	pending	pending
5003	Susie Ranalan	1pc. Apple iPad Mini	4/24/2024	2299.99	bank	pending	pending
5004	Anthon Ranalan	1pc. SONY WH-1000XM	4/24/2024	799.99	bank	success	dispatched

Figure 9: Order Management form (manage orders mode, admin view)

In comparison with customer view of the same form and the same mode, it features text fields and buttons for CRUD operations. Additionally, as admin, we are able to track ALL EXISTING orders within the database, unlike customer view which only tracks own orders from customer.

CODE – C#

C# is a versatile and modern programming language that is designed for building a wide range of applications on the .NET framework **(Jalli, 2022)**. A key factor of C# is that it features object-oriented programming; classes, objects, methods are utilized.

Figure 10: C#

For this project, C# was used to implement the coding part, with the IDE being Visual Studio 2022. All operations and queries of Customer file are only showcased, which sums up most CRUD operations, as every other files have similar codes. This is intended for summarizing purposes and to avoid lengthy reports.

ERROR HANDLING

In every method, error handling is implemented as we do not want any NULL or incorrect values to be inserted into the database.

```
if (checkAll() == "field and selection")
{
    MessageBox.Show("ERROR: record details are missing and no record is selected!");
    showCustomers();
}
else if (checkAll() == "selection")
{
    MessageBox.Show("ERROR: no record is selected!");
    showCustomers();
}
else if (checkAll() == "field")
{
    MessageBox.Show("ERROR: record details are missing!");
    showCustomers();
}
```

Figure 11: Code snippet of Customer check method

If it were to detect a blank field or empty selection, the user will receive a Message Box saying the error.

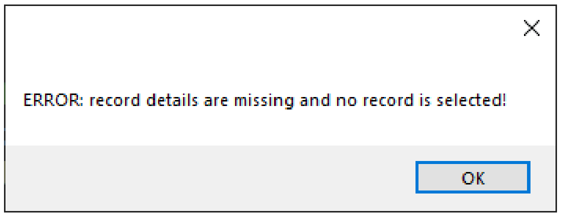

ERROR: record details are missing and no record is selected!

Figure 12: Message Box saying "ERROR: record details are missing and no record is selected!"

CREATE

```
private void createCustomer(CustomerDetails details)
{
    string query = "INSERT INTO Customer VALUES(@name, @email_address, @password, @phone_number,
@address_id);";

    try
    {
        SqlConnection connection = new SqlConnection(ConnectionString);
        connection.Open();

        int address_id = functions.createAddress(connection, details.street, details.city,
details.postal_code, details.country);

        SqlCommand command = new SqlCommand(query, connection);
        command.Parameters.AddWithValue("@name", details.name);
        command.Parameters.AddWithValue("@email_address", details.email_address);
        command.Parameters.AddWithValue("@password", details.password);
        command.Parameters.AddWithValue("@phone_number", details.phone_number);
        command.Parameters.AddWithValue("@address_id", address_id);
        command.ExecuteNonQuery();

        connection.Close();

        MessageBox.Show("SUCCESS: record added!");
    }

    catch (Exception x)
    {
        MessageBox.Show(x.Message);
    }
    showCustomers();
}
```

Figure 13: Code snippet of Customer create method

The basic workings of the create method is that it scans the text fields of the forms and it then inserts those values into the SQL query string which is then executed inside the database. Afterwards, it updates the table. If operation is successful, user will receive a message box that says “SUCCESS: record added!”. This method is then referenced in a WFA button, specifically “Add Customer” button.

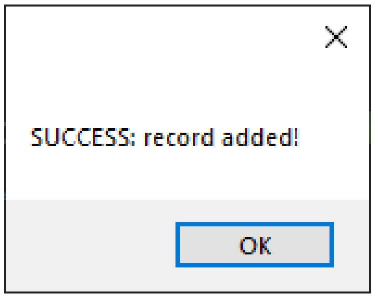

SUCCESS: record added!

Figure 14: Message box saying "SUCCESS: record added!"

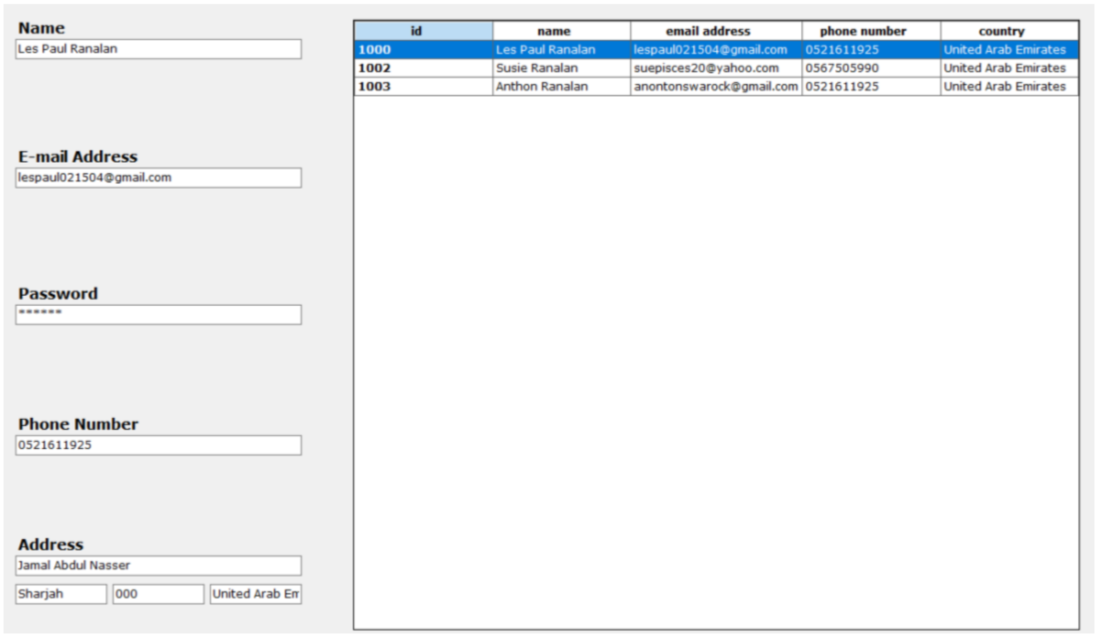
READ

```
private void readCustomer()
{
    if (c_LIST.SelectedRows.Count > 0)
    {
        DataGridViewRow selectedRow = c_LIST.SelectedRows[0];

        c_nameBOX.Text = selectedRow.Cells["name"].Value.ToString();
        c_emailBOX.Text = selectedRow.Cells["email_address"].Value.ToString();
        c_passwordBOX.Text = selectedRow.Cells["password"].Value.ToString();
        c_phoneBOX.Text = selectedRow.Cells["phone_number"].Value.ToString();
        c_streetBOX.Text = selectedRow.Cells["street"].Value.ToString();
        c_cityBOX.Text = selectedRow.Cells["city"].Value.ToString();
        c_postalBOX.Text = selectedRow.Cells["postal_code"].Value.ToString();
        c_countryBOX.Text = selectedRow.Cells["country"].Value.ToString();
    }
}
```

Figure 15: Code snippet of Customer read method

The basic workings of this method is that when user selects a row in the customer table, the data of that highlighted record is displayed on the text fields. This method is then referenced in a WFA trigger, specifically “SelecedRows” trigger. If operation is successful, text fields should change to the data of highlighted record.



Name	Les Paul Ranalan
E-mail Address	lespaul021504@gmail.com
Password	******
Phone Number	0521611925
Address	Jamal Abdul Nasser
	Sharjah
	000
	United Arab Em

id	name	email address	phone number	country
1000	Les Paul Ranalan	lespaul021504@gmail.com	0521611925	United Arab Emirates
1002	Susie Ranalan	suepisces20@yahoo.com	0567505990	United Arab Emirates
1003	Anthon Ranalan	anontonswarock@gmail.com	0521611925	United Arab Emirates

Figure 16: Text fields changed based on highlighted record

UPDATE

```
private void updateCustomer(int customer_id, int address_id, CustomerDetails details)
{
    string query = "UPDATE Customer SET name = @name, email_address = @email_address, password = 
@password, phone_number = @phone_number WHERE id = @customer_id";

    try
    {
        SqlConnection connection = new SqlConnection(ConnectionString);
        connection.Open();

        functions.updateAddress(connection, details.street, details.city, details.postal_code, 
details.country, address_id);

        SqlCommand command = new SqlCommand(query, connection);
        command.Parameters.AddWithValue("@name", details.name);
        command.Parameters.AddWithValue("@email_address", details.email_address);
        command.Parameters.AddWithValue("@password", details.password);
        command.Parameters.AddWithValue("@phone_number", details.phone_number);
        command.Parameters.AddWithValue("@customer_id", customer_id);
        command.ExecuteNonQuery();

        connection.Close();

        MessageBox.Show("SUCCESS: record updated!");
    }
    catch (Exception x)
    {
        MessageBox.Show(x.Message);
    }
    showCustomers();
}
```

Figure 17: Code snippet of Customer update method

The basic workings of this method is similar to create method wherein it scans the text fields and inserts it into the SQL query string. However, with read method, user can highlight a specific record, the text fields update, and user can now then edit any information. Once done, user clicks the Update button and if operation is successful, the user will receive a Message Box saying “SUCCESS: record updated!”. This method is then referenced in a WFA button, specifically “Update” button.

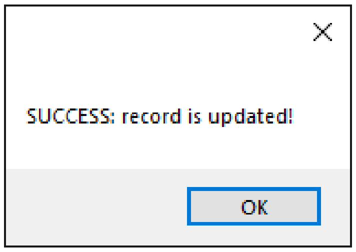

SUCCESS: record is updated!

Figure 18: Message Box saying "SUCCESS: record is updated!"

DELETE

```
private void deleteCustomer(int customer_id, int address_id)
{
    string query = "DELETE FROM Customer WHERE id = @customer_id; DELETE FROM Address WHERE id
= @address_id";

    try
    {
        SqlConnection connection = new SqlConnection(ConnectionString);

        connection.Open();

        SqlCommand command = new SqlCommand(query, connection);
        command.Parameters.AddWithValue("@customer_id", customer_id);
        command.Parameters.AddWithValue("@address_id", address_id);
        command.ExecuteNonQuery();

        MessageBox.Show("SUCCESS: record deleted!");
        showCustomers();

        connection.Close();
    }
    catch (Exception x)
    {
        MessageBox.Show(x.Message);
    }
}
```

Figure 19: Code snippet of Customer delete method

The basic workings of this method is similar to update method wherein it tracks a highlighted record. However, for delete method, the ID is only extracted from the highlighted record. That ID is then inserted into the SQL query string and then executed. If operation is successful, user will receive a Message Box saying “SUCCESS: record deleted!”. This method is then referenced in a WFA button, specifically “Delete” button.

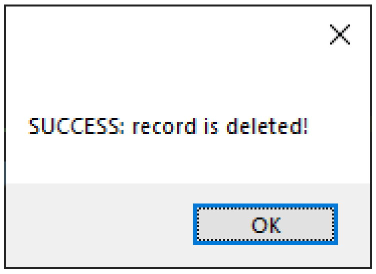

SUCCESS: record is deleted!

Figure 20: Message Box saying "SUCCESS: record is deleted!"

SEARCH

```
private void searchCustomer(string word)
{
    string query = "SELECT c.id AS id, c.name AS name, c.email_address AS email_address,
c.password AS password, c.phone_number AS phone_number, a.id AS address_id ,a.street AS
street, a.city AS city, a.postal_code AS postal_code, a.country AS country FROM Customer c
INNER JOIN Address a ON c.address_id = a.id WHERE c.id LIKE @word OR c.name LIKE @word OR
c.email_address LIKE @word OR c.phone_number LIKE @word OR a.country LIKE @word";

    try
    {
        SqlConnection connection = new SqlConnection(ConnectionString);
        connection.Open();

        SqlCommand command = new SqlCommand(query, connection);
        command.Parameters.AddWithValue("@word", "%" + word + "%");

        SqlDataAdapter adapter = new SqlDataAdapter(command);

        DataTable datatable = new DataTable();

        adapter.Fill(datatable);

        c_LIST.DataSource = datatable;

        connection.Close();
    }
    catch (Exception x)
    {
        MessageBox.Show(x.Message);
    }

}
```

Figure 21: Code snippet of Customer search method

The basic workings of the search method is that it basically uses the same query to get the current data from the database but it includes the “LIKE” query where it displays similar words as what the user is searching in specific columns. In this case, search method was specified on name, email address, phone number, and country columns. Once the record is found, it will display on the current DataGridView table. Otherwise, none will be displayed as the system found nothing.

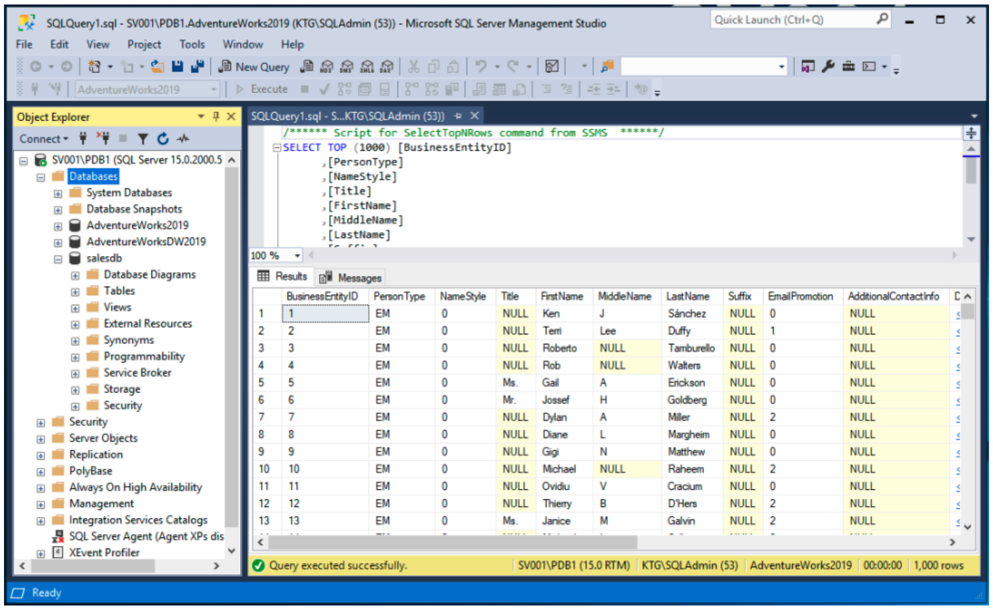
DATABASE – MICROSOFT SQL SERVER

Microsoft SQL Server is a powerful relational database management system (**RDMS**) that is widely used for storing and managing structured data **(Hughes, 2019)**. This enables the exploration of database operations and queries such as creating, reading, updating, and deleting (**CRUD**) of data records.

Figure 22: Microsoft SQL

Thus, it is the preferred option for this project because of its easy integration with Visual Studio 2022 and with any C# projects.

It also offers a GUI for managing databases easily, which is called Microsoft SQL Server Management Studio (**MSSMS**). This is where queries are executed.



```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [BusinessEntityID]
      ,[PersonType]
      ,[NameStyle]
      ,[Title]
      ,[FirstName]
      ,[MiddleName]
      ,[LastName]
```

	BusinessEntityID	PersonType	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	EmailPromotion	AdditionalContactInfo
1	1	EM	0	NULL	Ken	J	Sánchez	NULL	0	NULL
2	2	EM	0	NULL	Terri	Lee	Duffy	NULL	1	NULL
3	3	EM	0	NULL	Roberto	NULL	Tamburello	NULL	0	NULL
4	4	EM	0	NULL	Rob	NULL	Walters	NULL	0	NULL
5	5	EM	0	Ms.	Gail	A	Erickson	NULL	0	NULL
6	6	EM	0	Mr.	Jossef	H	Goldberg	NULL	0	NULL
7	7	EM	0	NULL	Dylan	A	Miller	NULL	2	NULL
8	8	EM	0	NULL	Diane	L	Margheim	NULL	0	NULL
9	9	EM	0	NULL	Gigi	N	Matthew	NULL	0	NULL
10	10	EM	0	NULL	Michael	NULL	Raheem	NULL	2	NULL
11	11	EM	0	NULL	Ovidiu	V	Cracium	NULL	0	NULL
12	12	EM	0	NULL	Thierry	B	D'Hers	NULL	2	NULL
13	13	EM	0	Ms.	Janice	M	Galvin	NULL	2	NULL

```
Query executed successfully. | SV001\PDB1 (15.0 RTM) | KTG\SQLAdmin (53) | AdventureWorks2019 | 00:00:00 | 1,000 rows
```

Figure 23: Microsoft SQL Server Management Studio

QUERIES

SQL queries were used to create a database and the tables inside said database. It features the same exact database schema as previously seen in the last assignment with little to no changes. Additionally, foreign key constraints were also implemented as a way to link tables between other tables. Below is an example code when creating Customer table.

```sql
CREATE TABLE Customer (
  id INT NOT NULL IDENTITY(1000,1) PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email_address VARCHAR(255) NOT NULL,
  [password] VARCHAR(255) NOT NULL,
  phone_number VARCHAR(20) NOT NULL,
  address_id INT NOT NULL,
  CONSTRAINT FK_Customer_Address FOREIGN KEY (address_id) REFERENCES
[Address] (id)
);
```

Figure 24: SQL Code for Customer table

This process is then repeated with other tables such as Address, Supplier, Staff, Product, Transaction, Order, and OrderDetail. All of which are linked together with foreign keys.

It is also worth mentioning that SQL was used to perform queries based on CRUD operations as seen in the previous chapters with commands such as UPDATE, INSERT, and DELETE.

REFACTORING

Refactoring is the process of restructuring existing code without changing its external behavior **(Kara, 2023)**. This is to improve readibility, maintainability, and performance. It is important to employ refactorization to eliminate “code smells” and ensures that the codebase is clean and concise.

Below are some concepts of refactorization that was applied onto the project.

(next page)

EXCTRACT METHOD

Extract method is where you take a section of a code that performs a specific task and move it into its own method.

BEFORE:

```
private void createCustomer(string name, string email_address, string password, string
phone_number, string address_line, string city, string postal_code, string country)
{
  string a_query = "INSERT INTO Address VALUES(@address_line, @city, @postal_code, @country);
SELECT SCOPE_IDENTITY();";

  string c_query = "INSERT INTO Customer VALUES(@name, @email_address, @password,
@phone_number, @address_id);";

  SqlConnection connection = new SqlConnection(ConnectionString);

  connection.Open();

  SqlCommand a_command = new SqlCommand(a_query, connection);

  a_command.Parameters.AddWithValue("@address_line", address_line);
  a_command.Parameters.AddWithValue("@city", city);
  a_command.Parameters.AddWithValue("@postal_code", postal_code);
  a_command.Parameters.AddWithValue("@country", country);

  int address_id = Convert.ToInt32(a_command.ExecuteScalar());

  SqlCommand c_command = new SqlCommand(c_query, connection);

  c_command.Parameters.AddWithValue("@name", name);
  c_command.Parameters.AddWithValue("@email_address", email_address);
  c_command.Parameters.AddWithValue("@password", password);
  c_command.Parameters.AddWithValue("@phone_number", phone_number);
  c_command.Parameters.AddWithValue("@address_id", address_id);

  c_command.ExecuteNonQuery();

  connection.Close();
}
```

Figure 25: Code snippet of Extract method BEFORE refactorization

AFTER:

```
public void createCustomer(string name, string email_address, string password, string
phone_number, string street, string city, string postal_code, string country)
{
  string query = "INSERT INTO Customer VALUES(@name, @email_address, @password, @phone_number,
@address_id);";

  SqlConnection connection = new SqlConnection(ConnectionString);
  connection.Open();

  int address_id = createAddress(connection, street, city, postal_code, country);

  SqlCommand command = new SqlCommand(query, connection);
  command.Parameters.AddWithValue("@name", name);
  command.Parameters.AddWithValue("@email_address", email_address);
  command.Parameters.AddWithValue("@password", password);
  command.Parameters.AddWithValue("@phone_number", phone_number);
  command.Parameters.AddWithValue("@address_id", address_id);

  command.ExecuteNonQuery();

  connection.Close();
}



public int createAddress(SqlConnection connection, string street, string city, string
postal_code, string country)
{
  string query = "INSERT INTO Address VALUES(@street, @city, @postal_code, @country); SELECT
SCOPE_IDENTITY();";

  SqlCommand command = new SqlCommand(query, connection);
  command.Parameters.AddWithValue("@street", street);
  command.Parameters.AddWithValue("@city", city);
  command.Parameters.AddWithValue("@postal_code", postal_code);
  command.Parameters.AddWithValue("@country", country);

  return Convert.ToInt32(command.ExecuteScalar());
}
```

Figure 26: Code snippet of Extract method AFTER refactorization

In the above example, the codes for adding address is now a separate method whereas before refactorization, it was combined in one method.

REPLACE MAGIC NUMBER WITH NAMED CONSTANT

BEFORE:

```
private void loginUser(string email_address, string password)
{
  SqlConnection connection = new SqlConnection("Data Source=(LocalDb)\\MSSQLLocalDB;Initial
Catalog=CircuitCentral;Integrated Security=True;Encrypt=False");
  connection.Open();

  int result = checkUser(connection, email_address, password);

  if (email_address == "admin" && password == "admin")
  {
    Constant.user_id = 0;
    functions.customerForm(this);
  }
  else
  {
    Constant.user_id = result;
    functions.productForm(this);
  }
  connection.Close();
}
```

Figure 27: Code snippet of Replace magic number with named constant BEFORE refactorization

AFTER:

```
public const string ConnectionString = "Data Source=(LocalDb)\\MSSQLLocalDB;Initial
Catalog=CircuitCentral;Integrated Security=True;Encrypt=False";
public const string adminEmail = "admin";
public const string adminPassword = "admin";

private void loginUser(string email_address, string password)
{
  SqlConnection connection = new SqlConnection(ConnectionString);
  connection.Open();

  int result = checkUser(connection, email_address, password);

  if (email_address == adminEmail && password == adminPassword)
  {
    Constant.user_id = 0;
    functions.customerForm(this);
  }
  else
  {
    Constant.user_id = result;
    functions.productForm(this);
  }
  connection.Close();
}
```

Figure 28: Code snippet of Rename variable/method AFTER refactorization

In the example above, a constant variable was created and applied to improve maintainability. If ever admin details were to change, we only need to change the constant variable’s value. Same goes for the Connection string.

REPLACE PARAMETERS WITH OBJECT

BEFORE:

```
public void createCustomer(string name, string email_address, string password, string phone_number, string
street, string city, string postal_code, string country)
{
    string query = "INSERT INTO Customer VALUES(@name, @email_address, @password, @phone_number, @address_id);";

    SqlConnection connection = new SqlConnection(ConnectionString);
    connection.Open();

    int address_id = createAddress(connection, street, city, postal_code, country);

    SqlCommand command = new SqlCommand(query, connection);
    command.Parameters.AddWithValue("@name", name);
    command.Parameters.AddWithValue("@email_address", email_address);
    command.Parameters.AddWithValue("@password", password);
    command.Parameters.AddWithValue("@phone_number", phone_number);
    command.Parameters.AddWithValue("@address_id", address_id);
    command.ExecuteNonQuery();

    connection.Close();
}
```

Figure 29: Code snippet of Replace parameters with object BEFORE refactorization

AFTER:

```
private void createCustomer(CustomerDetails details)
{
    string query = "INSERT INTO Customer VALUES(@name, @email_address, @password, @phone_number, @address_id);";

        SqlConnection connection = new SqlConnection(ConnectionString);
        connection.Open();

        int address_id = functions.createAddress(connection, details.street, details.city, details.postal_code,
details.country);

        SqlCommand command = new SqlCommand(query, connection);
        command.Parameters.AddWithValue("@name", details.name);
        command.Parameters.AddWithValue("@email_address", details.email_address);
        command.Parameters.AddWithValue("@password", details.password);
        command.Parameters.AddWithValue("@phone_number", details.phone_number);
        command.Parameters.AddWithValue("@address_id", address_id);
        command.ExecuteNonQuery();

        connection.Close();
}
```

Figure 30: Code snippet of Replace parameters with object AFTER refactorization

In the example above, a parameter object was utilized. Inside that object are its attributes such as customer.name, customer.email, etc. This is then used to reduce the amount of parameters that are passing inside a method, which looks cleaner and easier to understand.

REFLECTION

There were challeneges that were faced during the creation of the project, specifically executing SQL queries via C#. There is a bug wherein if the user tries to update an entity details, the system would throw out a foreign key exception.

An example to this would be deleting a customer record. In that scenario, when a customer record is deleted, its address record that is associated to that customer should also be deleted. The SQL commands were in correct sequence; delete address first, then customer record. This is quite strange as the query is executed in the correct sequence. This is also in the same scenario as the order records, wherein it has foreign keys to various entities such as orderdetail, transaction, and customer.

Alas, this problem was solved temporarily by enabling cascading deletion via the database query. Although, this is not recommended as a permanent fix as there is a higher chance for data loss.

Later on, the problem got solved by itself without the use of cascading deletion. Hence why this is a bug within the SQL execution.

GITHUB

GitHub is a web-based platform for version control via **Git (Korbin Brown, 2016)**. It facilitates collaboration among developers by providing features such as tracking changes, branch management, merging of code, etc.

Figure 31: GitHub

The project utilizes GitHub repoistories as a way of versoin control and tracking every changes that are made to the ccodebase. Below is the link of the repository.

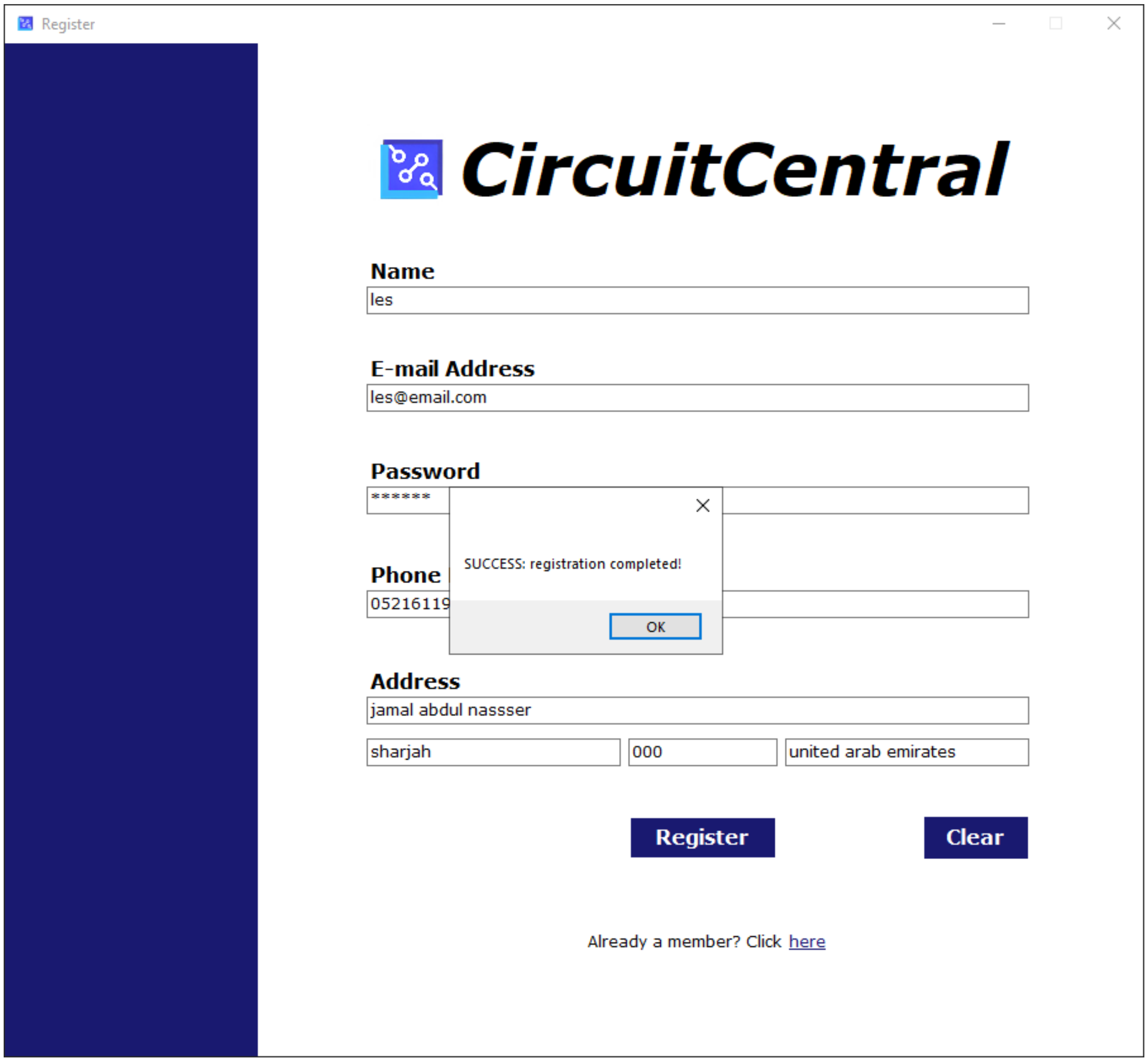
REPOSITORY

Below is the link to the public GitHub repostiroy, along with the readme file displayed.

https://github.com/shadowisf/CircuitCentral

EVIDENCE APPLICATION

REGISTER


Register

CircuitCentral

Name
les

E-mail Address
les@email.com

Password
******

Phone
05216119

SUCCESS: registration completed!

Address
jamal abdul nassser
sharjah
000
united arab emirates

Already a member? Click here

Figure 32: Registering on CircuitCentral

In the example above, registering will add a customer record named “les” with the necessary details.

LOGIN

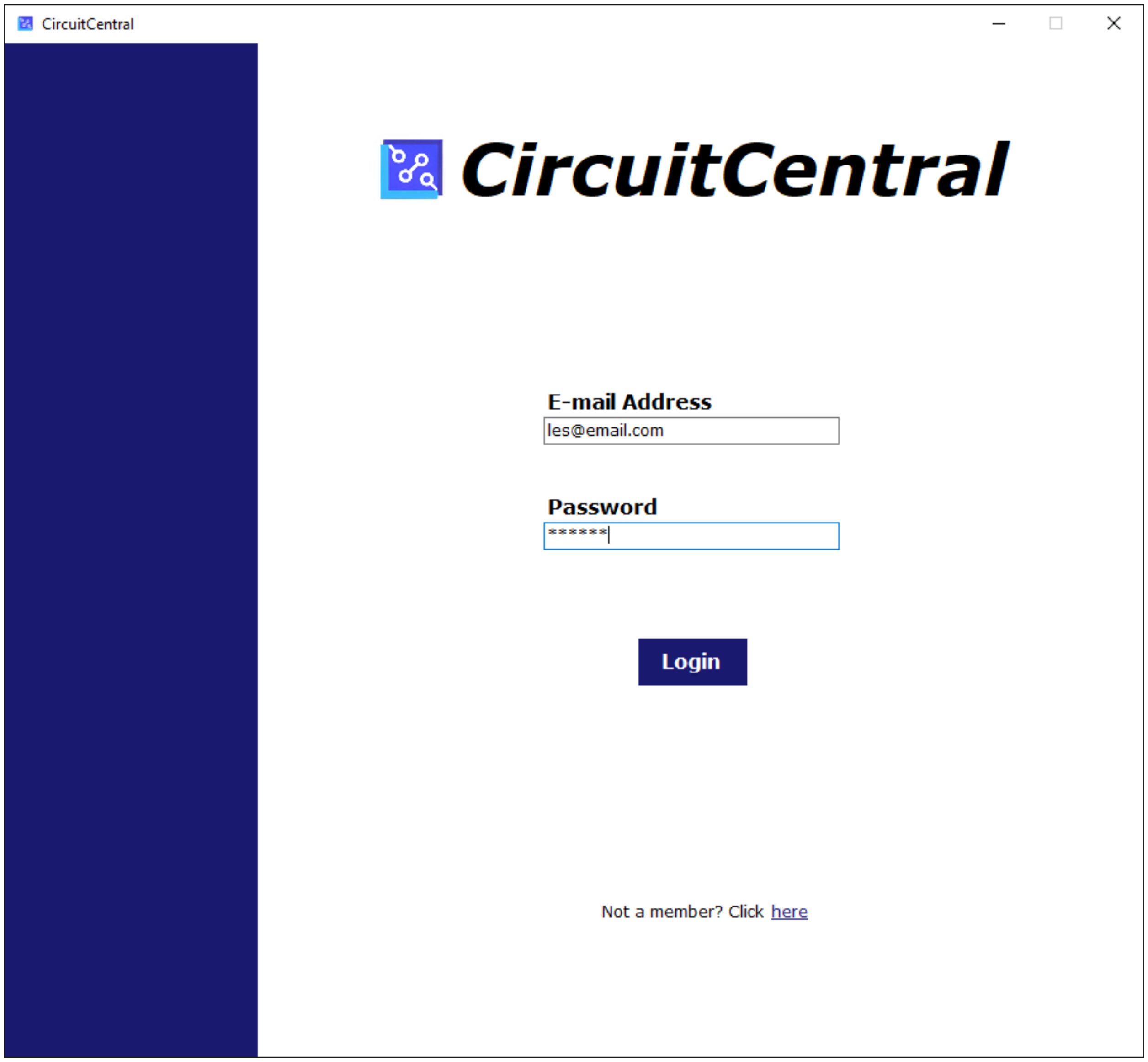


Figure 33: Logging in on CircuitCentral as Customer

Using the registration on the previous example, we are able to login via email address and password.

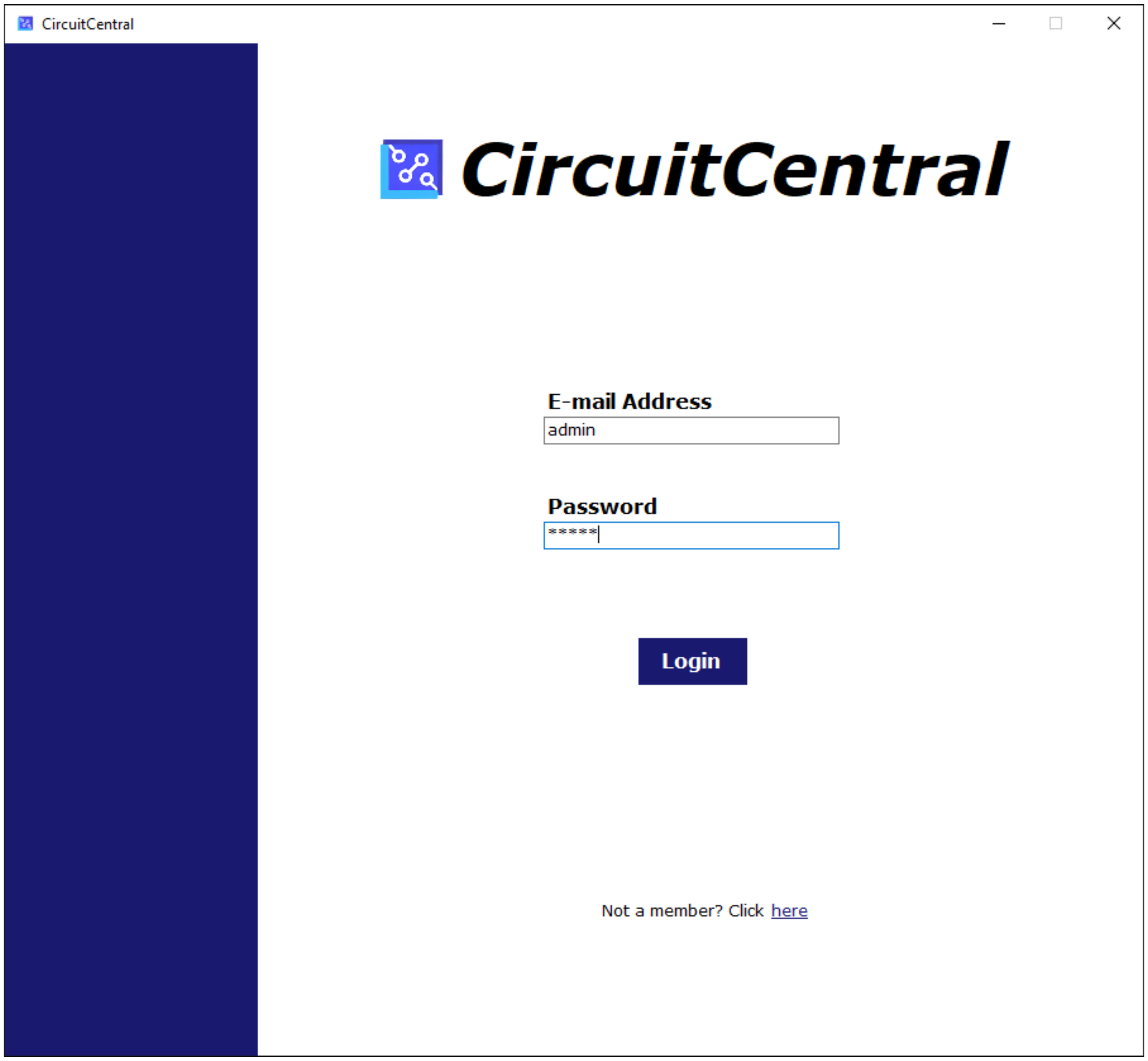
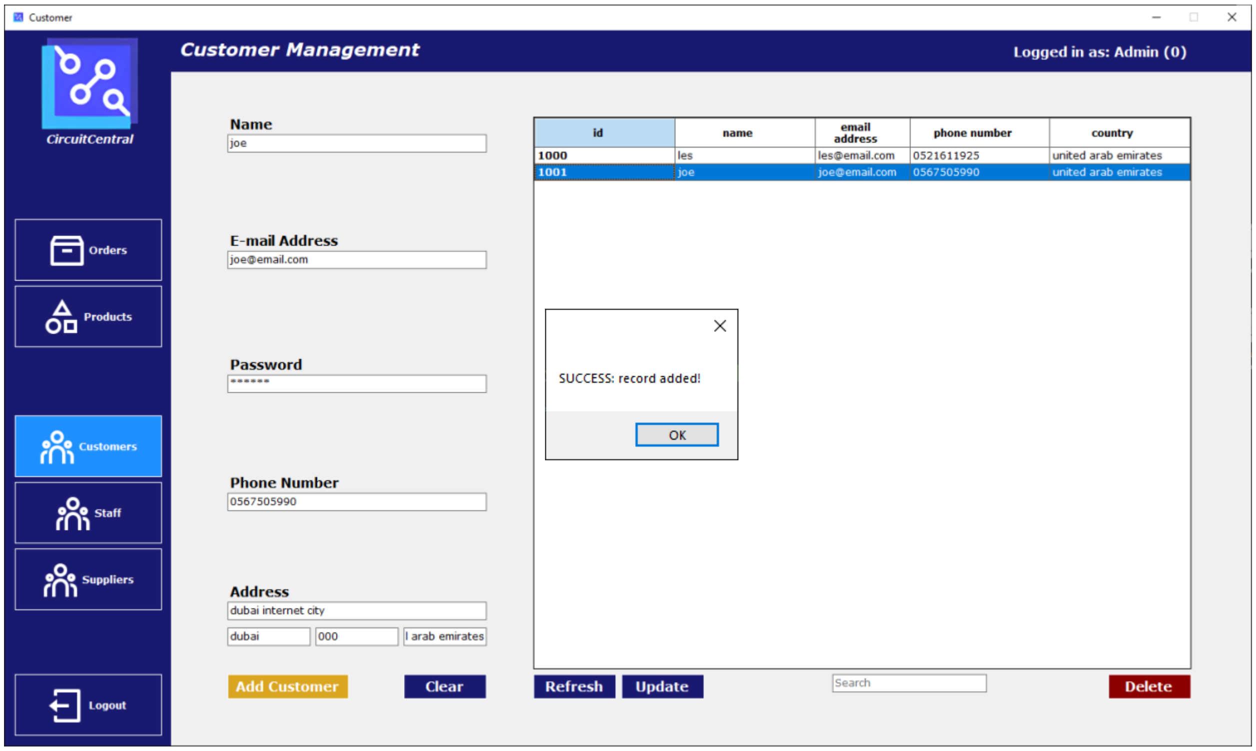


Figure 34: Logging in as admin

It is also possible to login with admin priviledges using “admin” as the email address and password.

CUSTOMER



Customer

Customer Management

Logged in as: Admin (0)

Name: joe

E-mail Address: joe@email.com

Password: ******

Phone Number: 0567505990

Address: dubai internet city

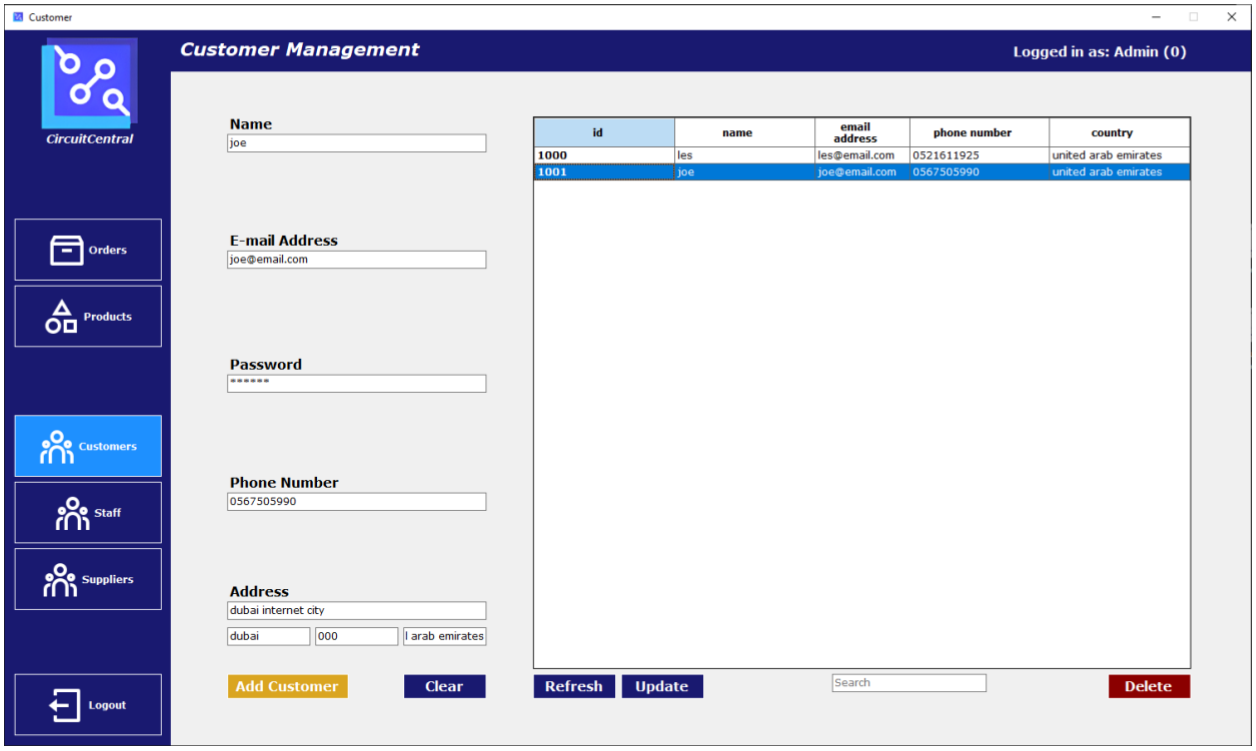
dubai | 000 | l arab emirates

id	name	email address	phone number	country
1000	les	les@email.com	0521611925	united arab emirates
1001	joe	joe@email.com	0567505990	united arab emirates

SUCCESS: record added!

Figure 35: Creating customer record

In the screenshot above, a new customer named “joe” was added. This is done by typing in information in the text fields and clicking the add customer button.



Name: joe

E-mail Address: joe@email.com

Password: ******

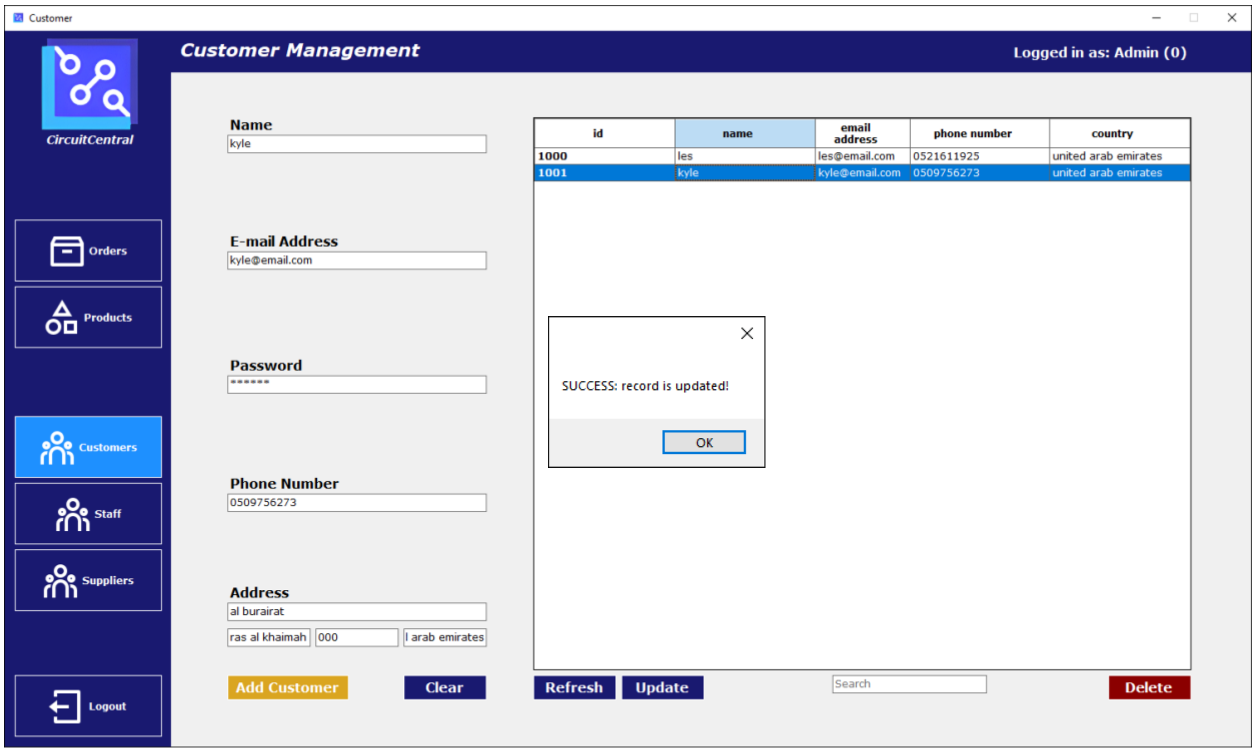
Phone Number: 0567505990

Address: dubai internet city | dubai | 000 | l arab emirates

id	name	email address	phone number	country
1000	les	les@email.com	0521611925	united arab emirates
1001	joe	joe@email.com	0567505990	united arab emirates

Figure 36: Reading customer record

When a row is highlighted in the table, it reads the data by setting it on the text fields.



Name: kyle

E-mail Address: kyle@email.com

Password: ******

Phone Number: 0509756273

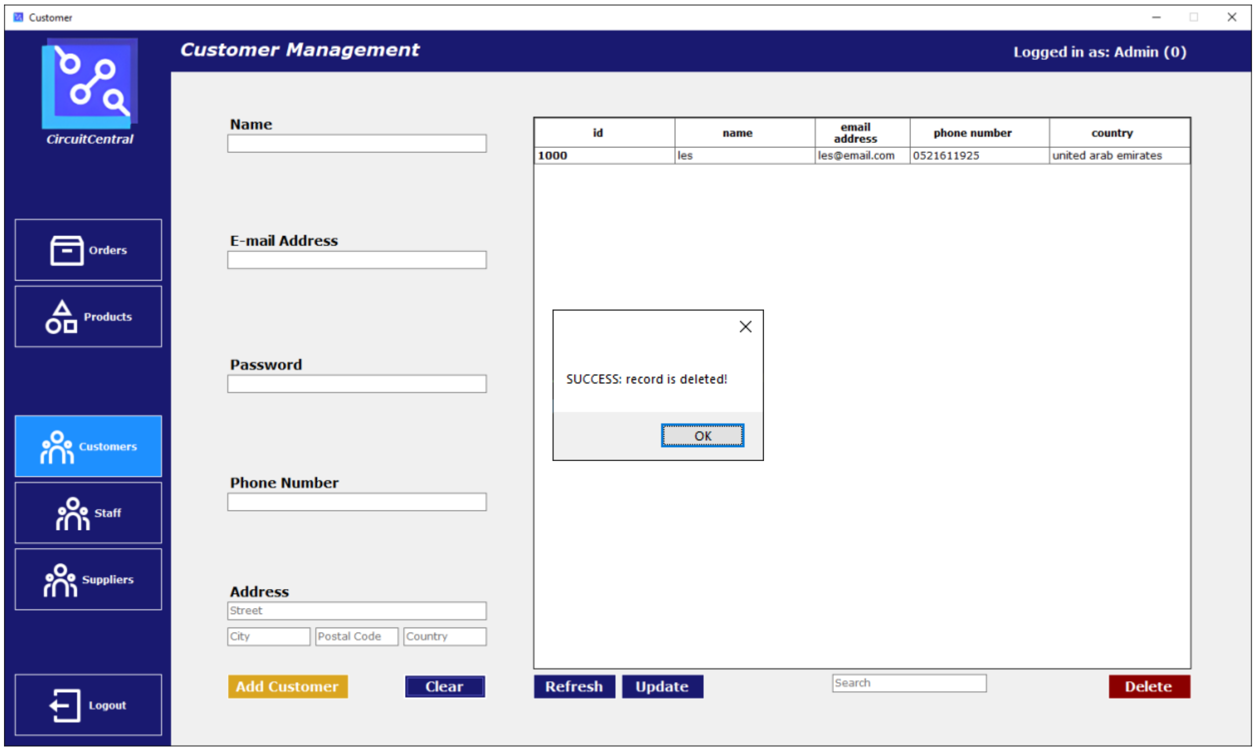
Address: al burairat | ras al khaimah | 000 | l arab emirates

id	name	email address	phone number	country
1000	les	les@email.com	0521611925	united arab emirates
1001	kyle	kyle@email.com	0509756273	united arab emirates

SUCCESS: record is updated!

Figure 37: Updating customer record

Using the previous example, we updated Customer 1001, which was previously Joe, to Kyle with different information. This is done by selecting a row, typing the updated information in the text fields, and finally pressing the update button.

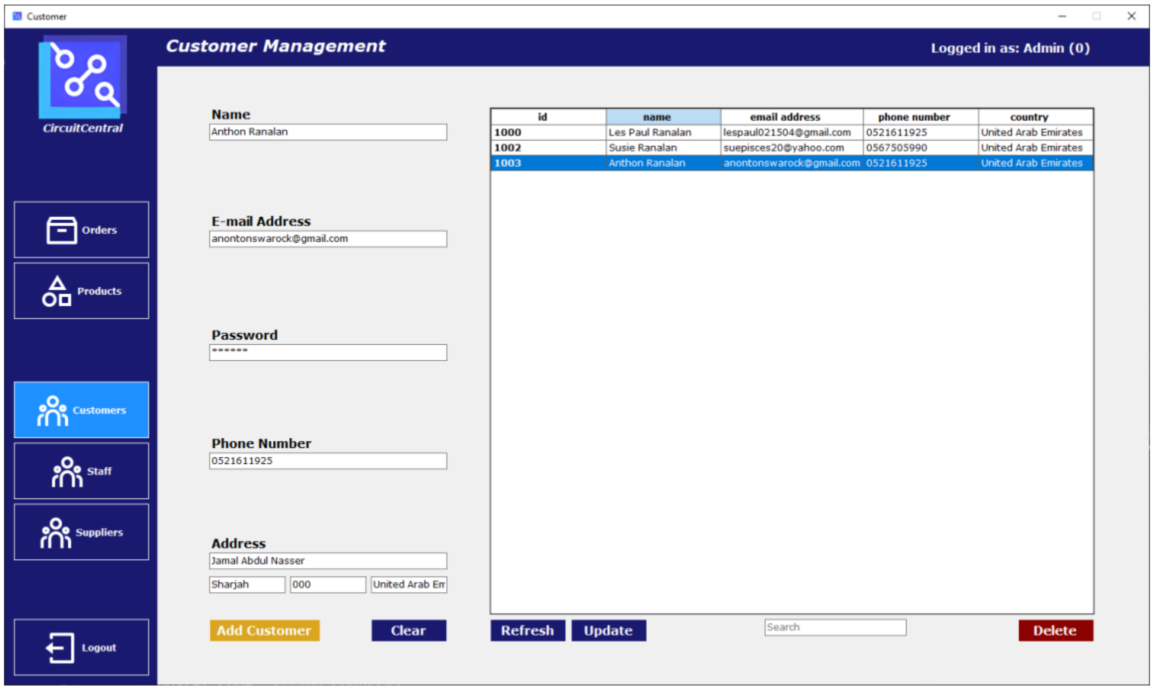


id	name	email address	phone number	country
1000	les	les@email.com	0521611925	united arab emirates

SUCCESS: record is deleted!

Figure 38: Deleting customer record

Using the previous example, Customer 1001 named Kyle was deleted. This is done by selecting a row and then clicking the delete button.



Name: Anthon Ranalan

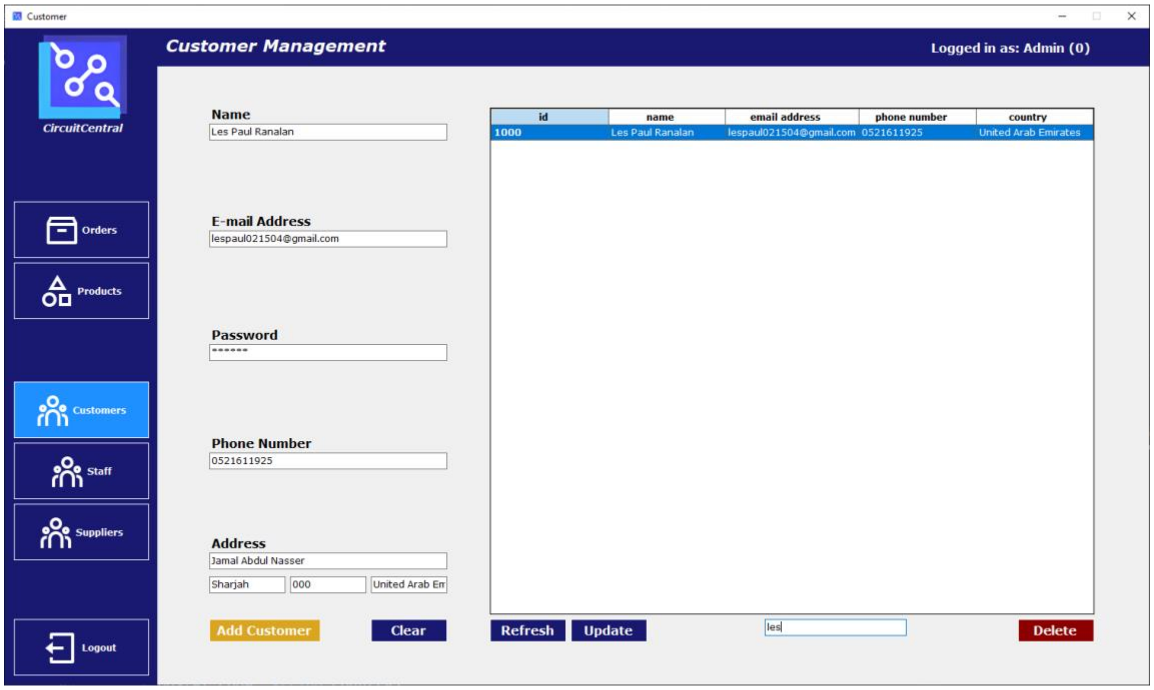
E-mail Address: anontonswarock@gmail.com

Password: ******

Phone Number: 0521611925

Address: Jamal Abdul Nasser | Sharjah | 000 | United Arab Em

id	name	email address	phone number	country
1000	Les Paul Ranalan	lespaul021504@gmail.com	0521611925	United Arab Emirates
1002	Susie Ranalan	suepisces20@yahoo.com	0567505990	United Arab Emirates
1003	Anthon Ranalan	anontonswarock@gmail.com	0521611925	United Arab Emirates



Name: Les Paul Ranalan

E-mail Address: lespaul021504@gmail.com

Password: ******

Phone Number: 0521611925

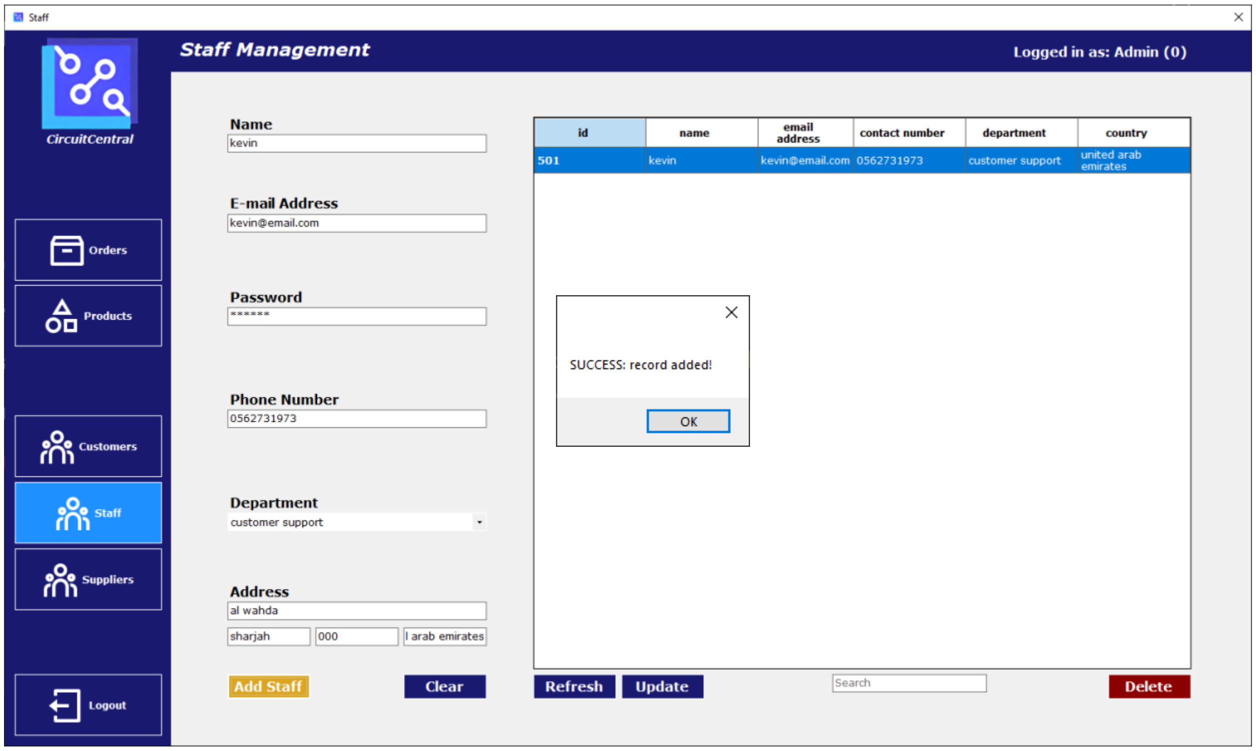
Address: Jamal Abdul Nasser | Sharjah | 000 | United Arab Em

id	name	email address	phone number	country
1000	Les Paul Ranalan	lespaul021504@gmail.com	0521611925	United Arab Emirates

Figure 39: Searching customer record

In the example above, the first screenshot shows all the customer records that are present within the database. We searched for the customer record “les”, which correctly returned one record.

STAFF



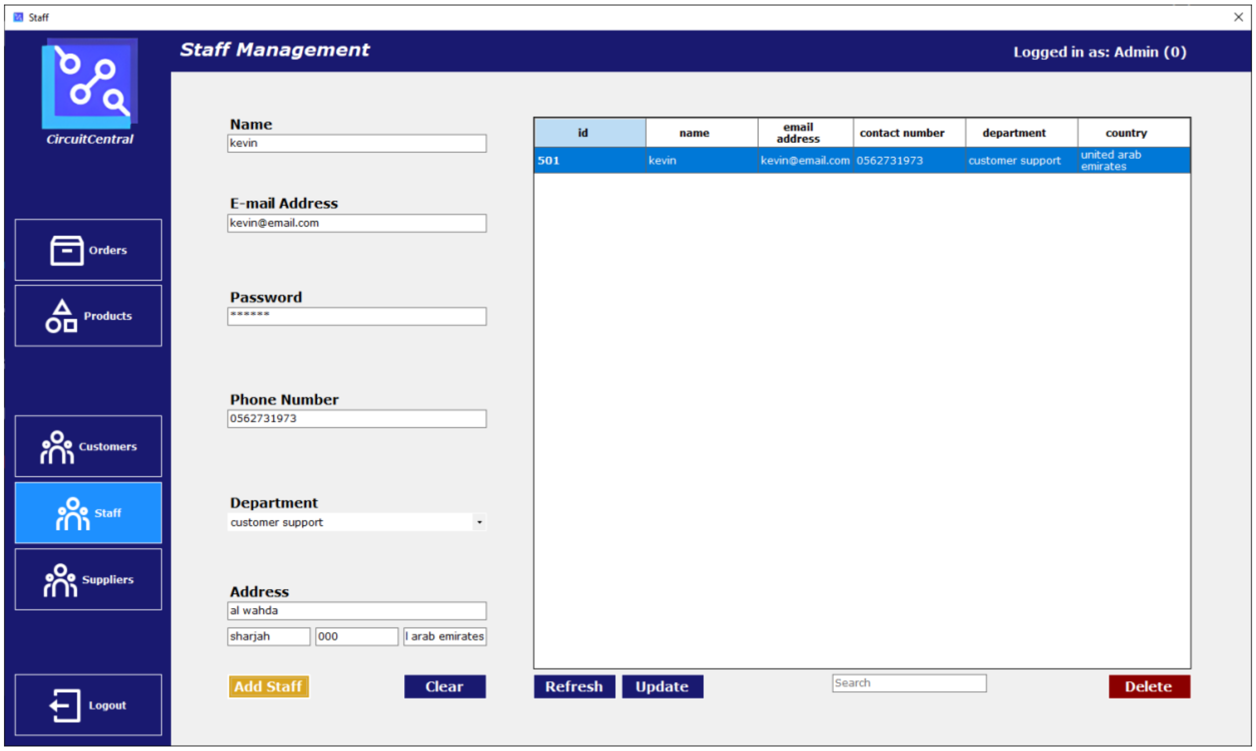
Name: kevin
E-mail Address: kevin@email.com
Password: ******
Phone Number: 0562731973
Department: customer support
Address: al wahda
sharjah | 000 | l arab emirates

id	name	email address	contact number	department	country
501	kevin	kevin@email.com	0562731973	customer support	united arab emirates

SUCCESS: record added!

Figure 40: Creating staff record

In the example above, we created a new staff named “Kevin”. This is done by entering information in the text fields and then clicking the add staff button.



Name: kevin

E-mail Address: kevin@email.com

Password: ******

Phone Number: 0562731973

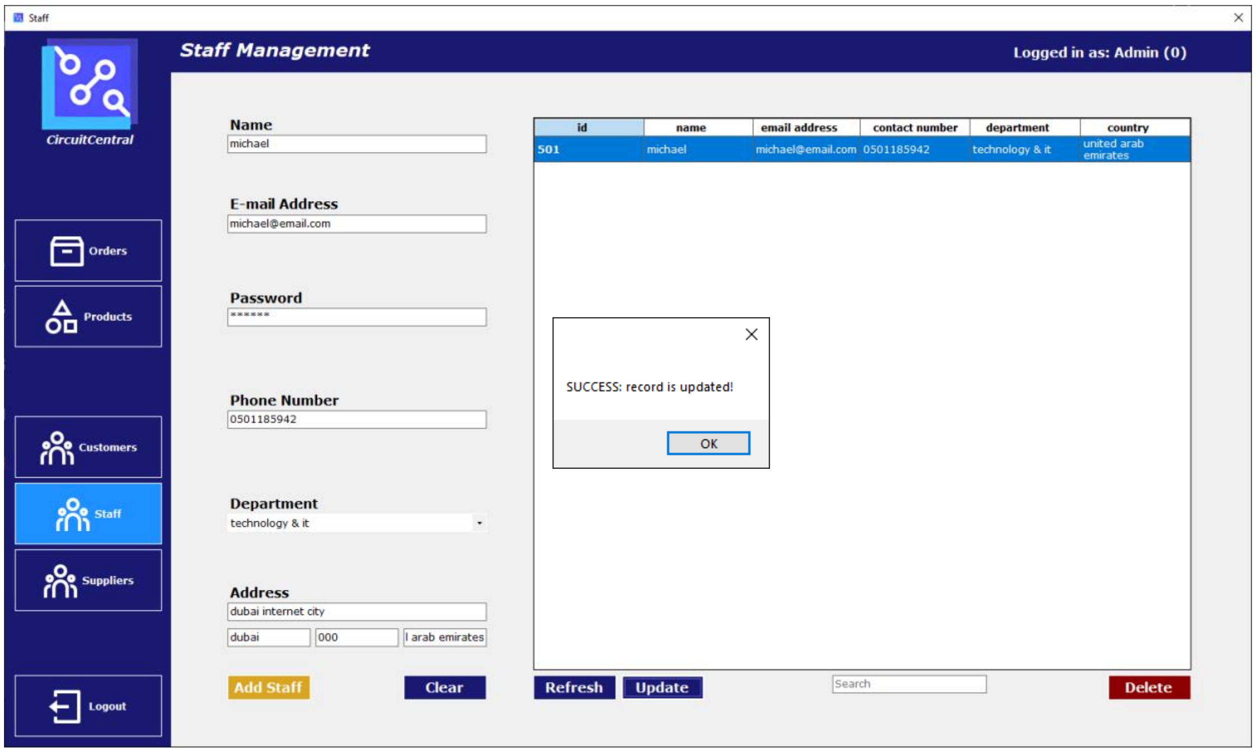
Department: customer support

Address: al wahda | sharjah | 000 | l arab emirates

id	name	email address	contact number	department	country
501	kevin	kevin@email.com	0562731973	customer support	united arab emirates

Figure 41: Reading staff record

When a row is highlighted in the table, it reads the data by setting it on the text fields.



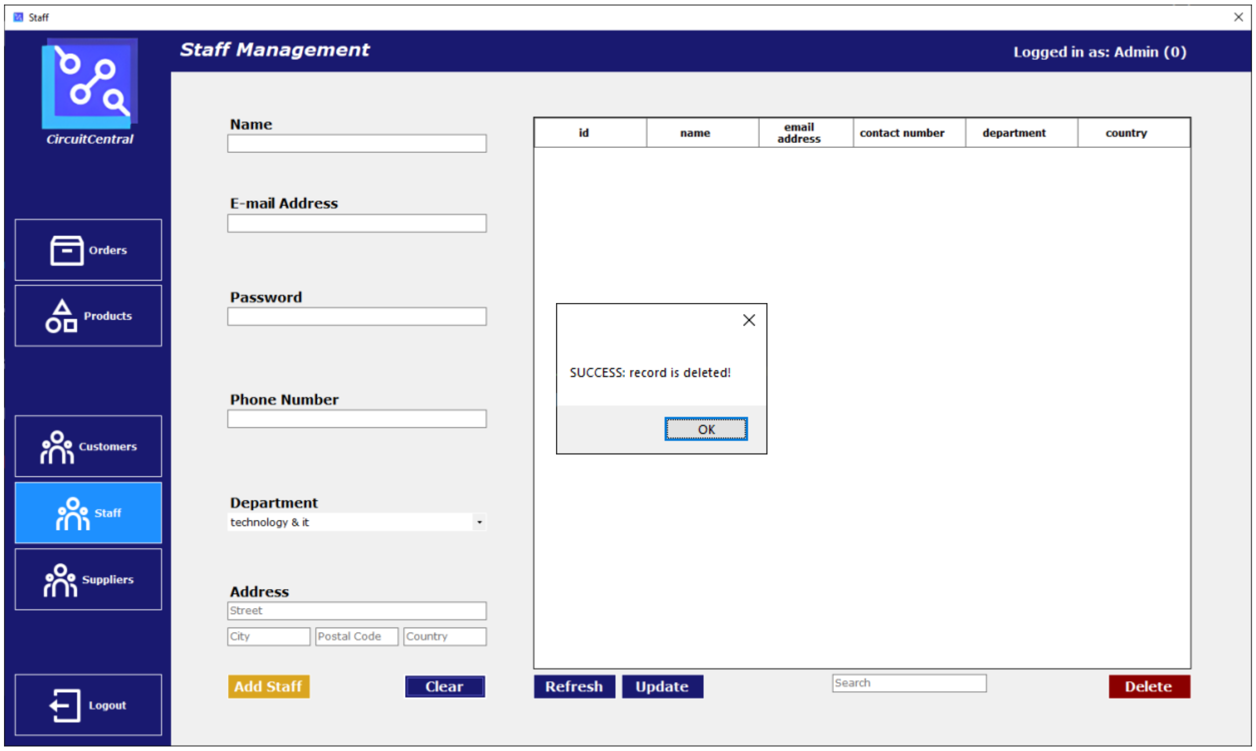
Name	michael
E-mail Address	michael@email.com
Password	******
Phone Number	0501185942
Department	technology & it
Address	dubai internet city
	dubai
	000
	l arab emirates

id	name	email address	contact number	department	country
501	michael	michael@email.com	0501185942	technology & it	united arab emirates

SUCCESS: record is updated!

Figure 42: Updating staff record

Using the previous example, we updated Staff 501, which was previously named "Kevin", to "Michael" with new information. This is done by highlighting a row, typing the new information in the text field and clicking the update button.

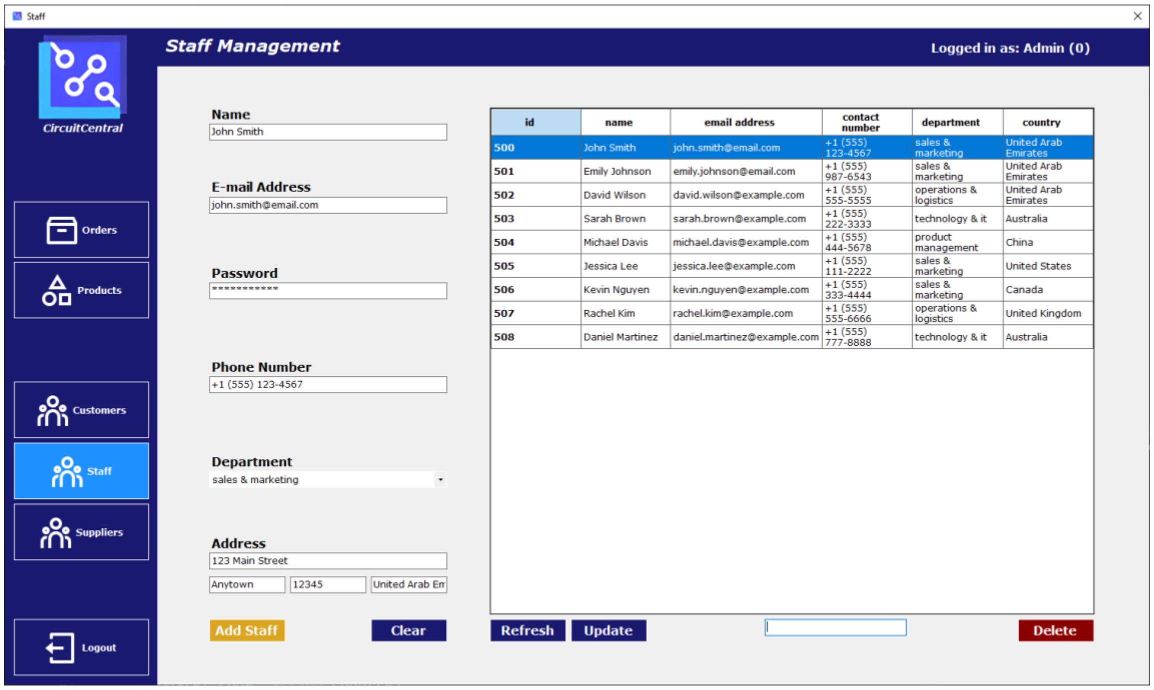


id	name	email address	contact number	department	country

SUCCESS: record is deleted!

Figure 43: Deleting staff record

Using the previous example, Staff 501 named Michael was deleted. This is done by highlighting the row and then clicking the delete button.



Logged in as: Admin (0)

Name: John Smith

E-mail Address: john.smith@email.com

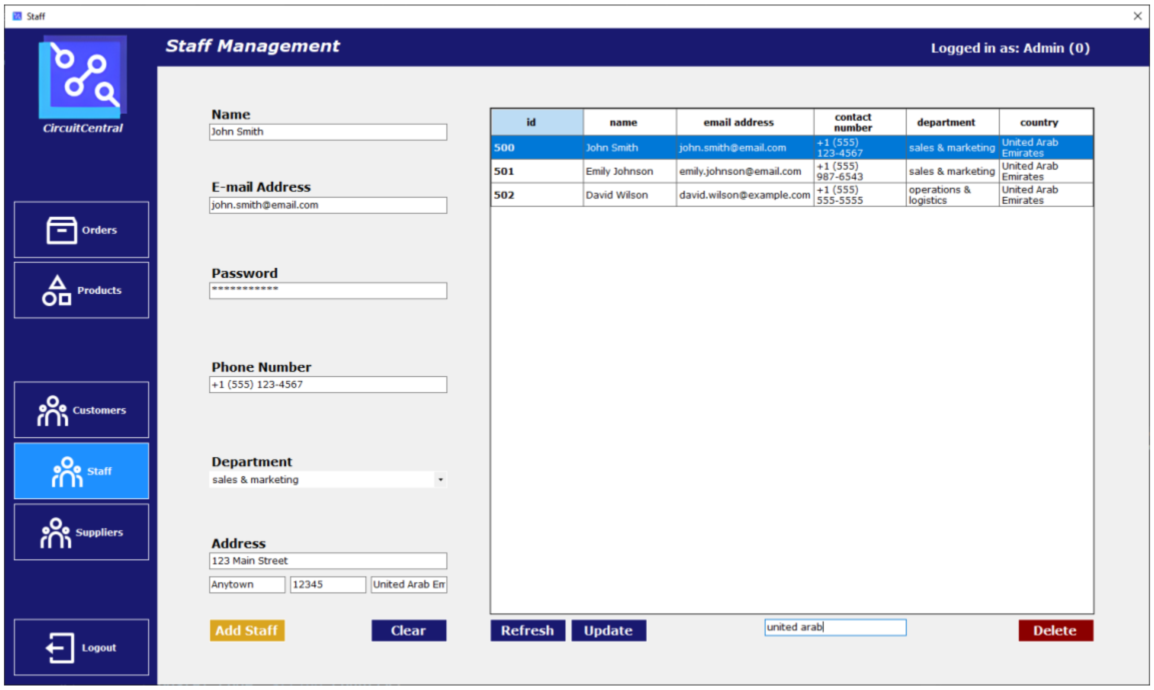
Password: ***********

Phone Number: +1 (555) 123-4567

Department: sales & marketing

Address: 123 Main Street | Anytown | 12345 | United Arab Em

id	name	email address	contact number	department	country
500	John Smith	john.smith@email.com	+1 (555) 123-4567	sales & marketing	United Arab Emirates
501	Emily Johnson	emily.johnson@email.com	+1 (555) 987-6543	sales & marketing	United Arab Emirates
502	David Wilson	david.wilson@example.com	+1 (555) 555-5555	operations & logistics	United Arab Emirates
503	Sarah Brown	sarah.brown@example.com	+1 (555) 222-3333	technology & it	Australia
504	Michael Davis	michael.davis@example.com	+1 (555) 444-5678	product management	China
505	Jessica Lee	jessica.lee@example.com	+1 (555) 111-2222	sales & marketing	United States
506	Kevin Nguyen	kevin.nguyen@example.com	+1 (555) 333-4444	sales & marketing	Canada
507	Rachel Kim	rachel.kim@example.com	+1 (555) 555-6666	operations & logistics	United Kingdom
508	Daniel Martinez	daniel.martinez@example.com	+1 (555) 777-8888	technology & it	Australia



Logged in as: Admin (0)

Name: John Smith

E-mail Address: john.smith@email.com

Password: ***********

Phone Number: +1 (555) 123-4567

Department: sales & marketing

Address: 123 Main Street | Anytown | 12345 | United Arab Em

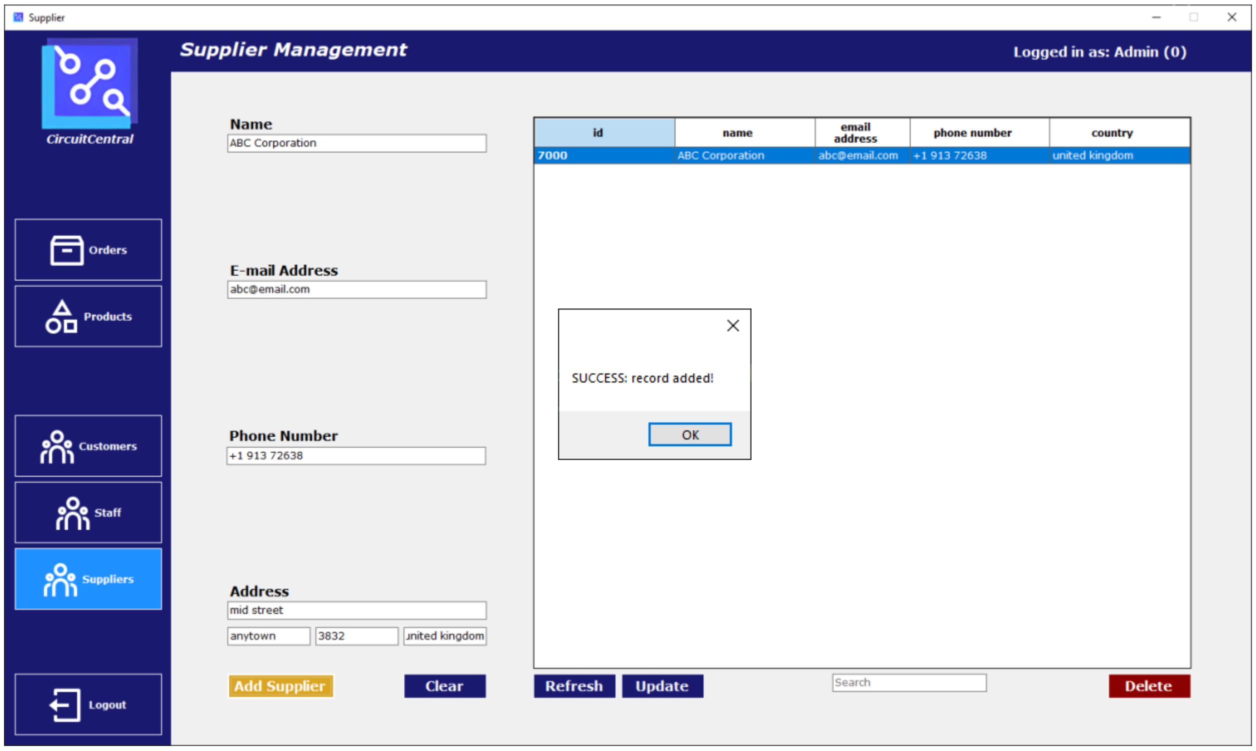
united arab

id	name	email address	contact number	department	country
500	John Smith	john.smith@email.com	+1 (555) 123-4567	sales & marketing	United Arab Emirates
501	Emily Johnson	emily.johnson@email.com	+1 (555) 987-6543	sales & marketing	United Arab Emirates
502	David Wilson	david.wilson@example.com	+1 (555) 555-5555	operations & logistics	United Arab Emirates

Figure 44: Searching staff record

In the example above, the first screenshot shows all of the staff records that are present within the database. We then searched for “united arab” which correctly returned three records that have country as “United Arab Emirates”.

SUPPLIER



Name: ABC Corporation

E-mail Address: abc@email.com

Phone Number: +1 913 72638

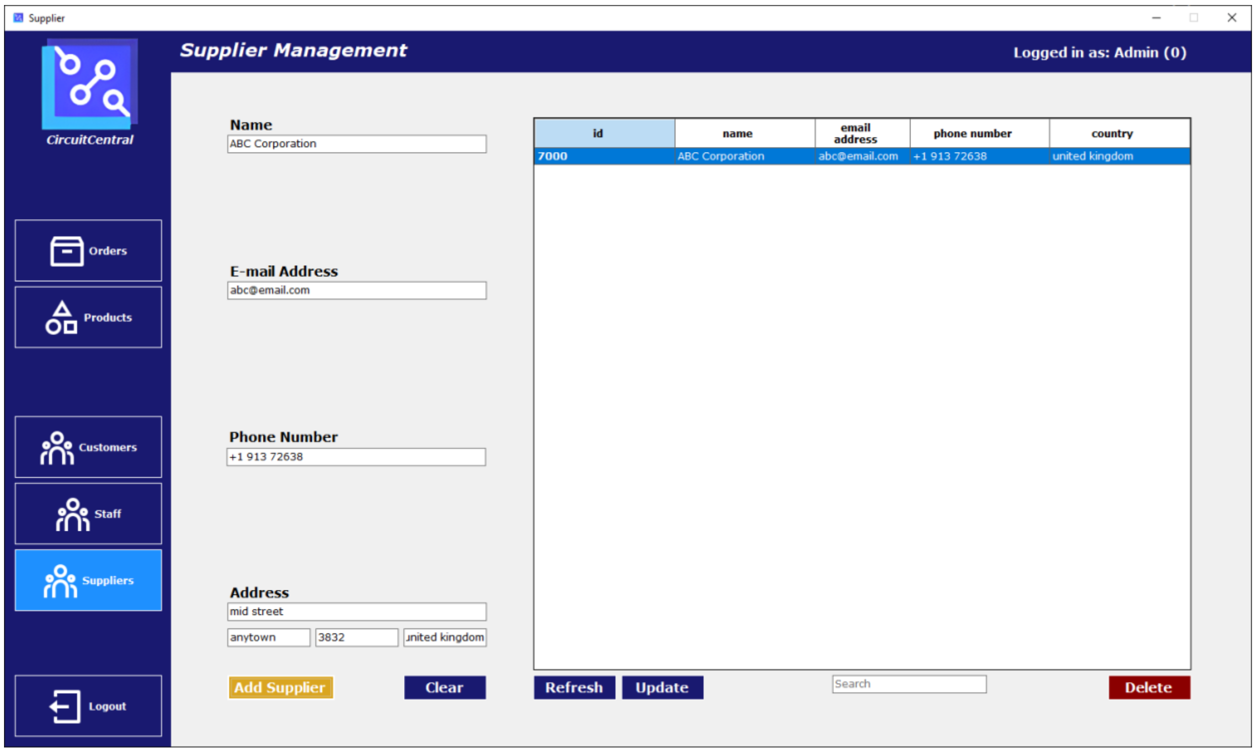
Address: mid street | anytown | 3832 | united kingdom

id	name	email address	phone number	country
7000	ABC Corporation	abc@email.com	+1 913 72638	united kingdom

SUCCESS: record added!

Figure 45: Adding supplier record

In the example above, we created a new Supplier record with the name of “ABC Corporation”. This is done by entering the necessary information within the text fields and clicking the add supplier button.



Name: ABC Corporation

E-mail Address: abc@email.com

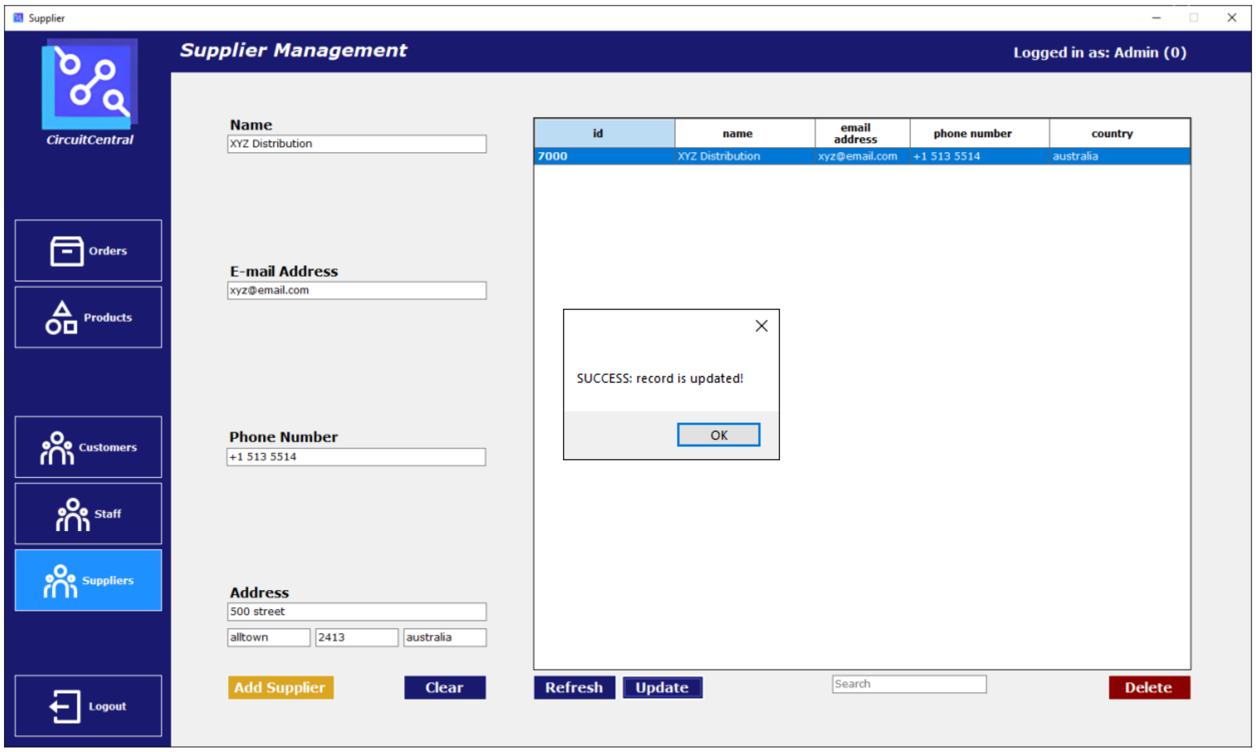
Phone Number: +1 913 72638

Address: mid street | anytown | 3832 | united kingdom

id	name	email address	phone number	country
7000	ABC Corporation	abc@email.com	+1 913 72638	united kingdom

Figure 46: Reading supplier record

When a row is highlighted in the table, it reads the data by setting it on the text fields.



Name: XYZ Distribution

E-mail Address: xyz@email.com

Phone Number: +1 513 5514

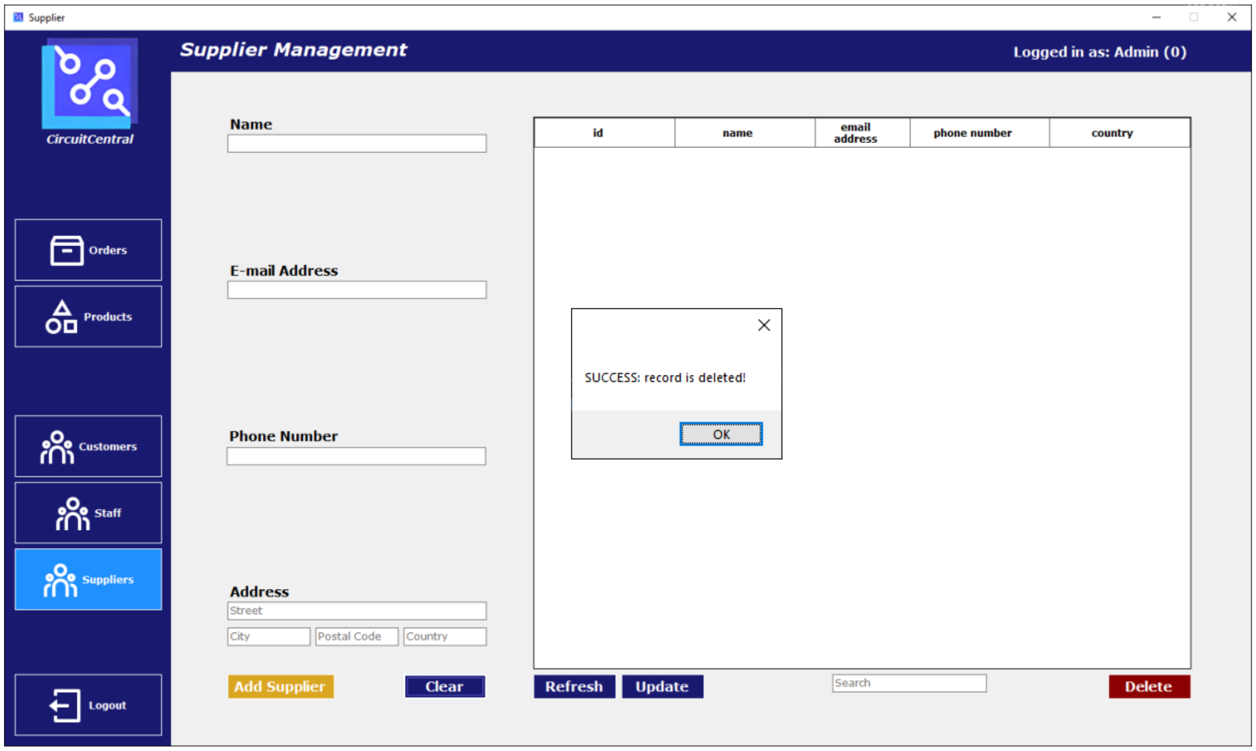
Address: 500 street | alltown | 2413 | australia

id	name	email address	phone number	country
7000	XYZ Distribution	xyz@email.com	+1 513 5514	australia

SUCCESS: record is updated!

Figure 47: Updating supplier record

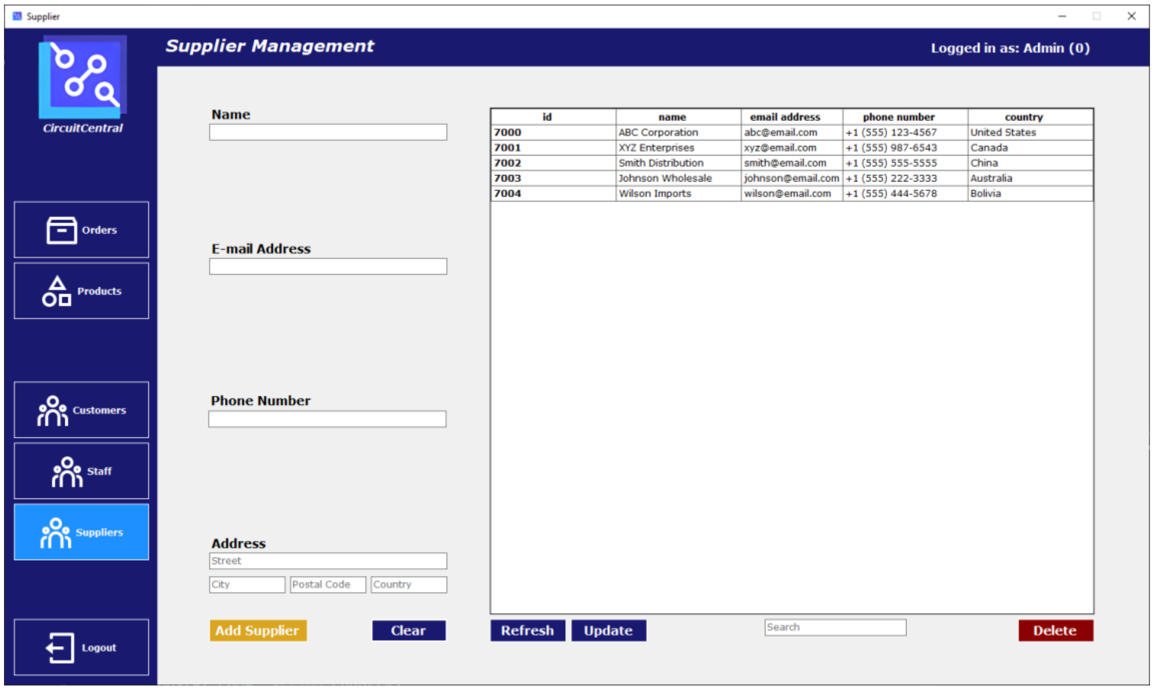
Using the previous example, we updated the details of Supplier 7000, which was previously named “ABC Corporation”, to have new details with the name of “XYZ Distribution”. This is done by highlighting the row, then enter the updated details into the text fields, and click the update button.



SUCCESS: record is deleted!

Figure 48: Deleting supplier record

Using the previous example, we deleted Supplier 7000. This is done by highlighting the row and the clicking the delete button.



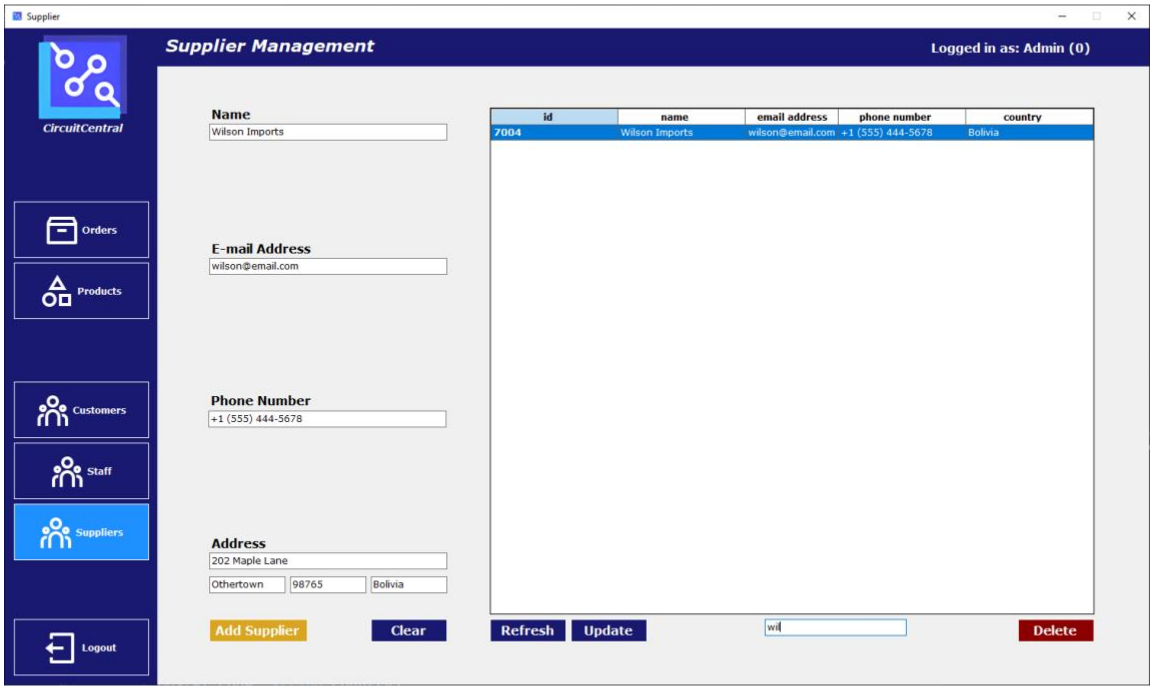
Name

E-mail Address

Phone Number

Address
Street
City | Postal Code | Country

id	name	email address	phone number	country
7000	ABC Corporation	abc@email.com	+1 (555) 123-4567	United States
7001	XYZ Enterprises	xyz@email.com	+1 (555) 987-6543	Canada
7002	Smith Distribution	smith@email.com	+1 (555) 555-5555	China
7003	Johnson Wholesale	johnson@email.com	+1 (555) 222-3333	Australia
7004	Wilson Imports	wilson@email.com	+1 (555) 444-5678	Bolivia



Name: Wilson Imports

E-mail Address: wilson@email.com

Phone Number: +1 (555) 444-5678

Address: 202 Maple Lane
Othertown | 98765 | Bolivia

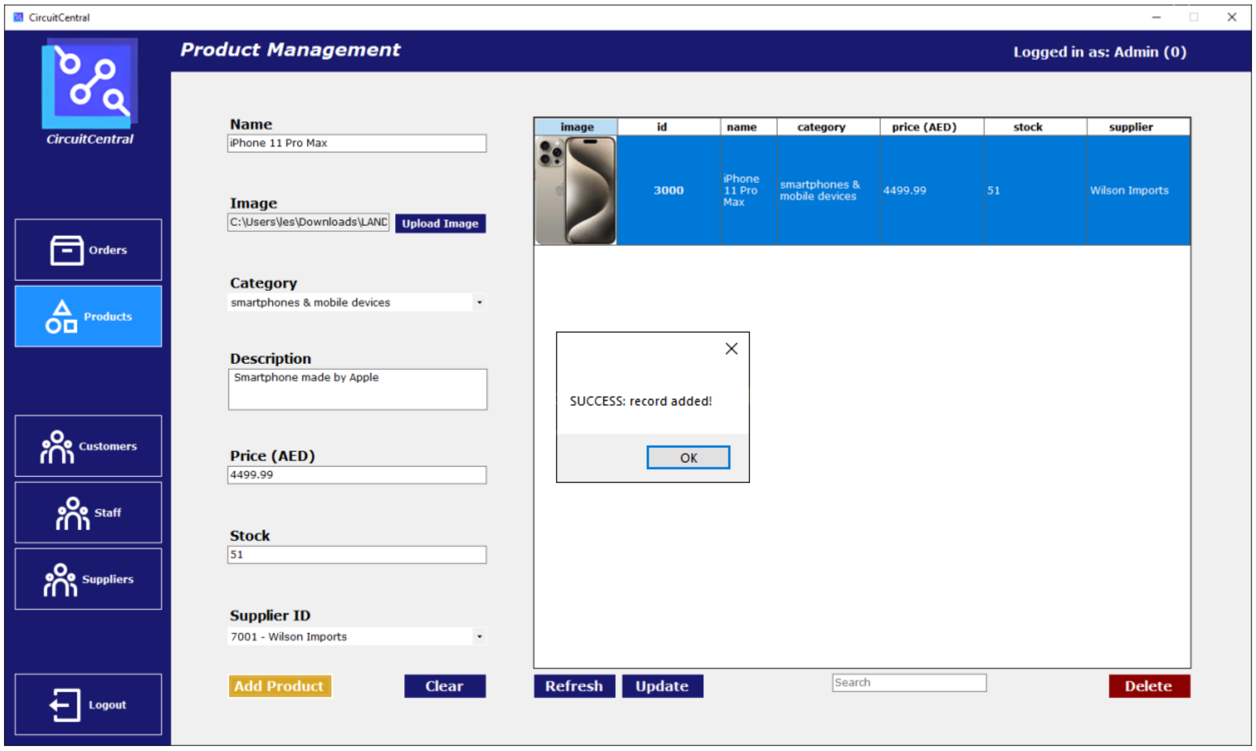
Search: wil

id	name	email address	phone number	country
7004	Wilson Imports	wilson@email.com	+1 (555) 444-5678	Bolivia

Figure 49: Searching supplier record

In the example above, the first screenshot shows all of the supplier records that are present within the database. We then searched for “wil” and it correcltly returned one record with the name as “Wilson Imports”.

PRODUCT


CircuitCentral

Product Management

Logged in as: Admin (0)

Name: iPhone 11 Pro Max

Image: C:\Users\es\Downloads\LANC

Category: smartphones & mobile devices

Description: Smartphone made by Apple

Price (AED): 4499.99

Stock: 51

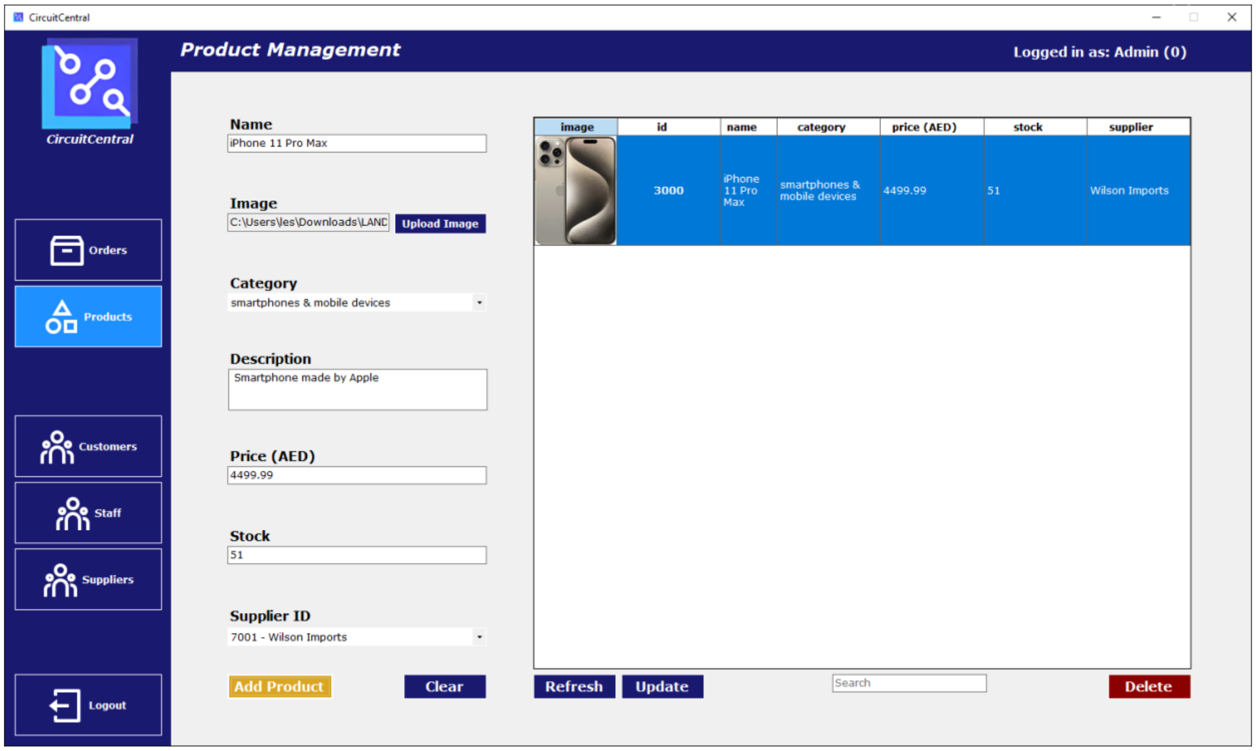
Supplier ID: 7001 - Wilson Imports

image	id	name	category	price (AED)	stock	supplier
	3000	iPhone 11 Pro Max	smartphones & mobile devices	4499.99	51	Wilson Imports

SUCCESS: record added!

Figure 50: Adding product record

In the example above, a product was added with the name as “iPhone 15 Pro Max”. This is done by entering the necessary information in the text fields and clicking the add product button.



Name: iPhone 11 Pro Max

Image: C:\Users\es\Downloads\LAND

Category: smartphones & mobile devices

Description: Smartphone made by Apple

Price (AED): 4499.99

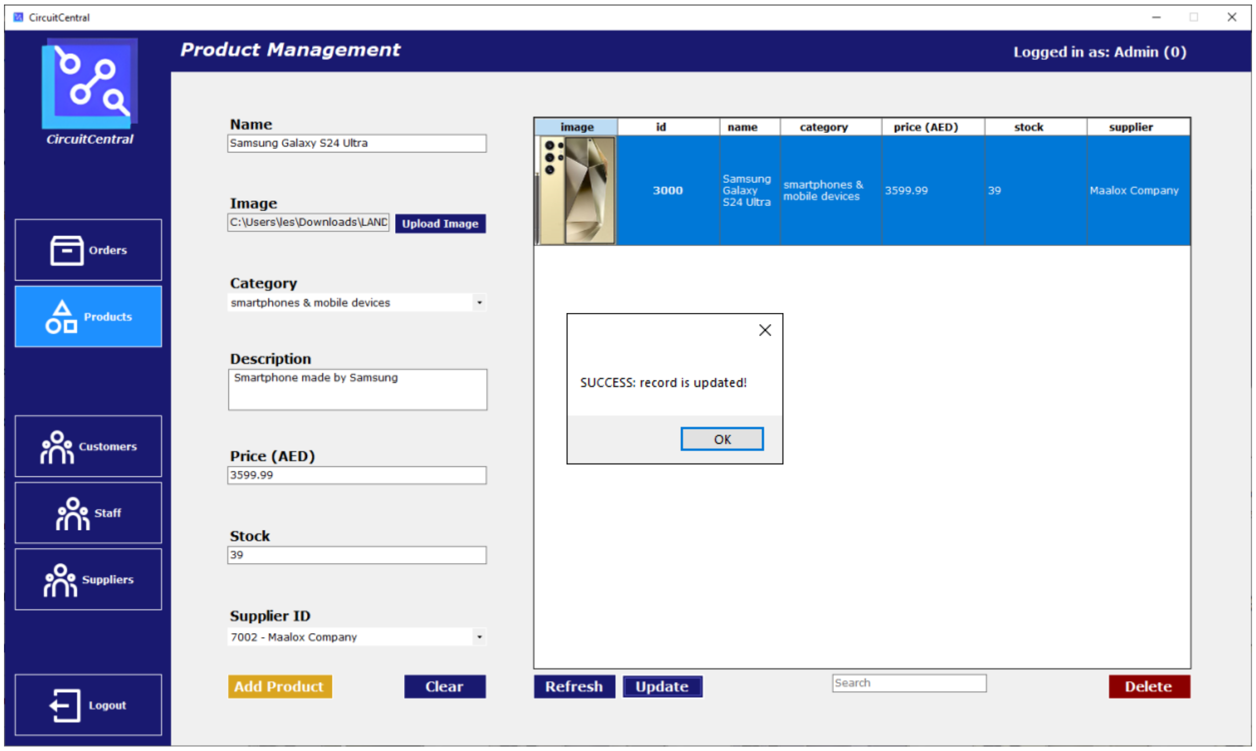
Stock: 51

Supplier ID: 7001 - Wilson Imports

image	id	name	category	price (AED)	stock	supplier
	3000	iPhone 11 Pro Max	smartphones & mobile devices	4499.99	51	Wilson Imports

Figure 51: Reading product record

When a row is highlighted in the table, it reads the data by setting it on the text fields.


CircuitCentral

Product Management

Logged in as: Admin (0)

Name: Samsung Galaxy S24 Ultra

Image: C:\Users\les\Downloads\LANC

Category: smartphones & mobile devices

Description: Smartphone made by Samsung

Price (AED): 3599.99

Stock: 39

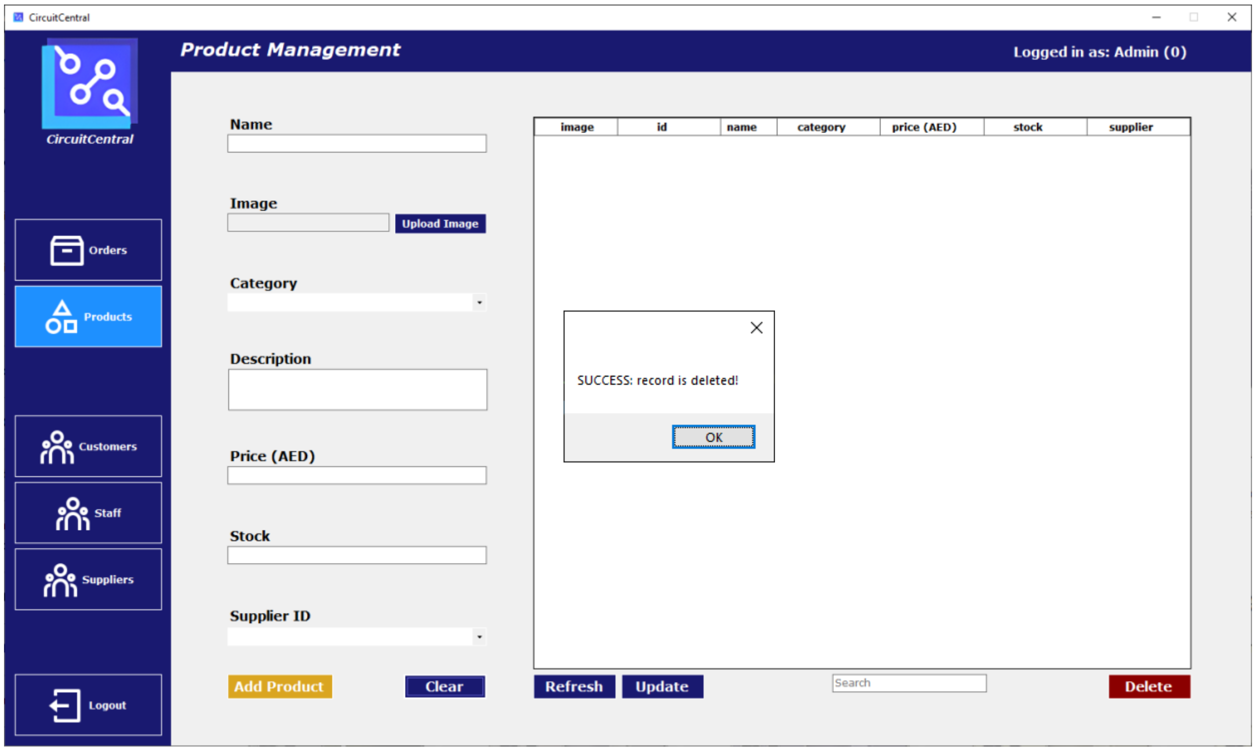
Supplier ID: 7002 - Maalox Company

image	id	name	category	price (AED)	stock	supplier
	3000	Samsung Galaxy S24 Ultra	smartphones & mobile devices	3599.99	39	Maalox Company

SUCCESS: record is updated!

Figure 52: Updating product record

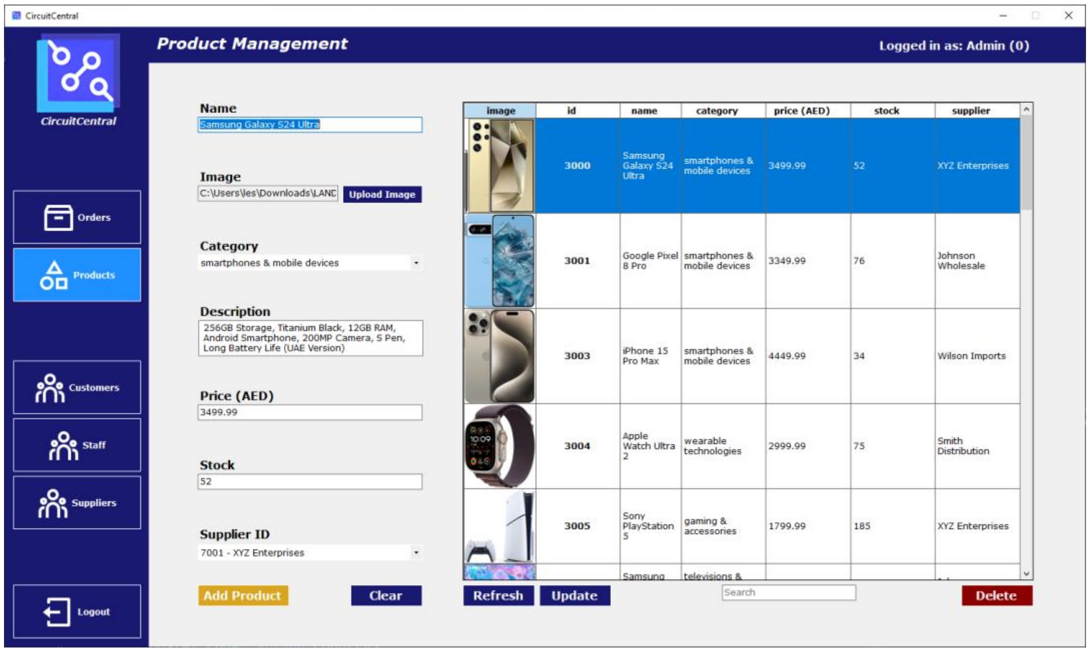
Using the previous example, we updated Product 3000, which was previously named "iPhone 15 Pro Max", into "Samsung Galaxy S24 Ultra" with new details along with it. This is done by highlighting a row, then entering the updated details on the text fields, and hitting the update button.



SUCCESS: record is deleted!

Figure 53: Deleting product record

Using the previous example, Product 3000 was deleted. This is done by highlighting a row and then clicking the delete button.



Name: Samsung Galaxy S24 Ultra

Image: C:\Users\les\Downloads\LANC

Category: smartphones & mobile devices

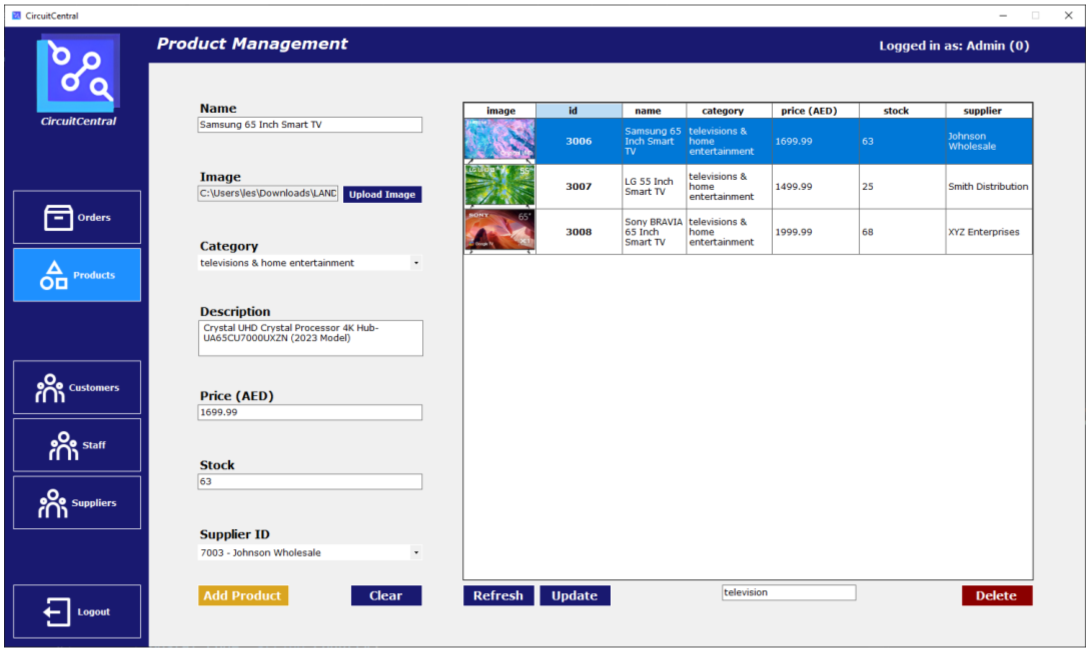
Description: 256GB Storage, Titanium Black, 12GB RAM, Android Smartphone, 200MP Camera, S Pen, Long Battery Life (UAE Version)

Price (AED): 3499.99

Stock: 52

Supplier ID: 7001 - XYZ Enterprises

image	id	name	category	price (AED)	stock	supplier
	3000	Samsung Galaxy S24 Ultra	smartphones & mobile devices	3499.99	52	XYZ Enterprises
	3001	Google Pixel 8 Pro	smartphones & mobile devices	3349.99	76	Johnson Wholesale
	3003	iPhone 15 Pro Max	smartphones & mobile devices	4449.99	34	Wilson Imports
	3004	Apple Watch Ultra 2	wearable technologies	2999.99	75	Smith Distribution
	3005	Sony PlayStation 5	gaming & accessories	1799.99	185	XYZ Enterprises
		Samsung	televisions &			



Name: Samsung 65 Inch Smart TV

Image: C:\Users\les\Downloads\LANC

Category: televisions & home entertainment

Description: Crystal UHD Crystal Processor 4K Hub-UA65CU7000UXZN (2023 Model)

Price (AED): 1699.99

Stock: 63

Supplier ID: 7003 - Johnson Wholesale

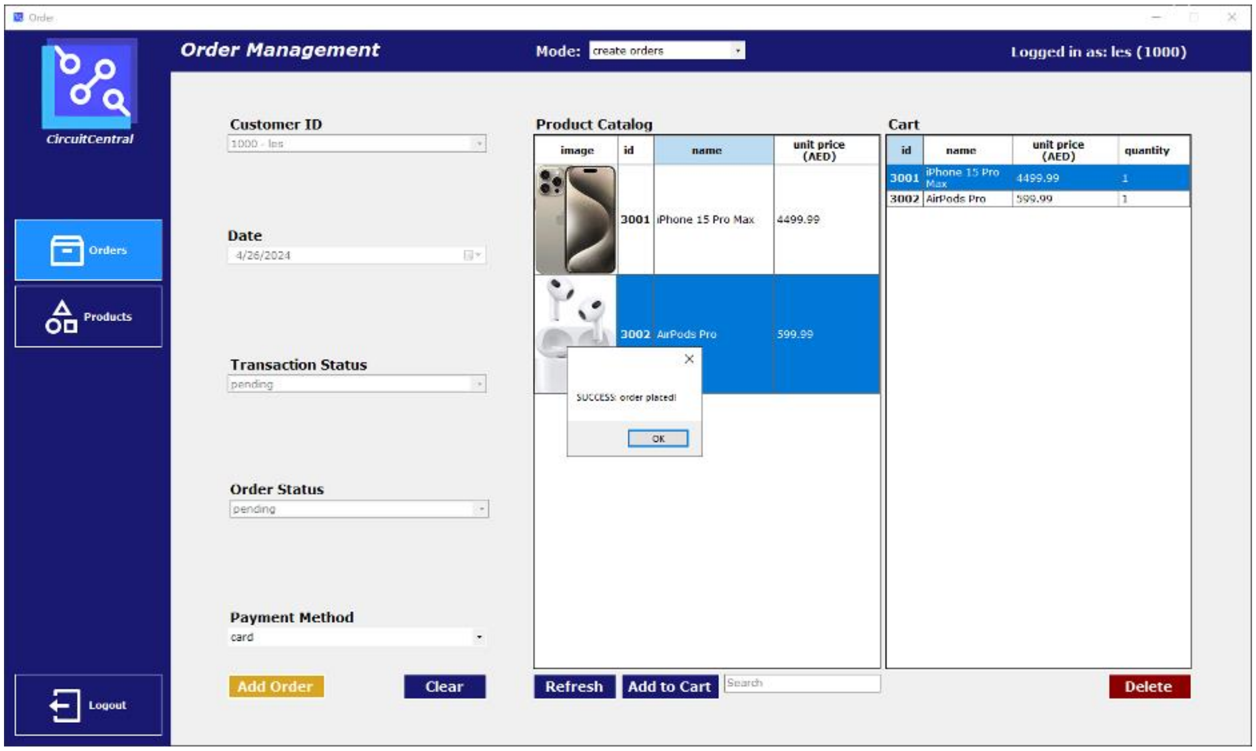
image	id	name	category	price (AED)	stock	supplier
	3006	Samsung 65 Inch Smart TV	televisions & home entertainment	1699.99	63	Johnson Wholesale
	3007	LG 55 Inch Smart TV	televisions & home entertainment	1499.99	25	Smith Distribution
	3008	Sony BRAVIA 65 Inch Smart TV	televisions & home entertainment	1999.99	68	XYZ Enterprises

television

Figure 54: Searching product record

In the example above, the first screenshot shows all the product records that are present within the database. We then searched for “television”, and it correctly returned three records, all with the category of “television & home entertainment”. Note that this is the same as Product Catalog within the Order Management form.

ORDER



Order Management

Mode: create orders

Logged in as: les (1000)

Customer ID: 1000 - les

Date: 4/26/2024

Transaction Status: pending

Order Status: pending

Payment Method: card

Product Catalog

image	id	name	unit price (AED)
	3001	iPhone 15 Pro Max	4499.99
	3002	AirPods Pro	599.99

Cart

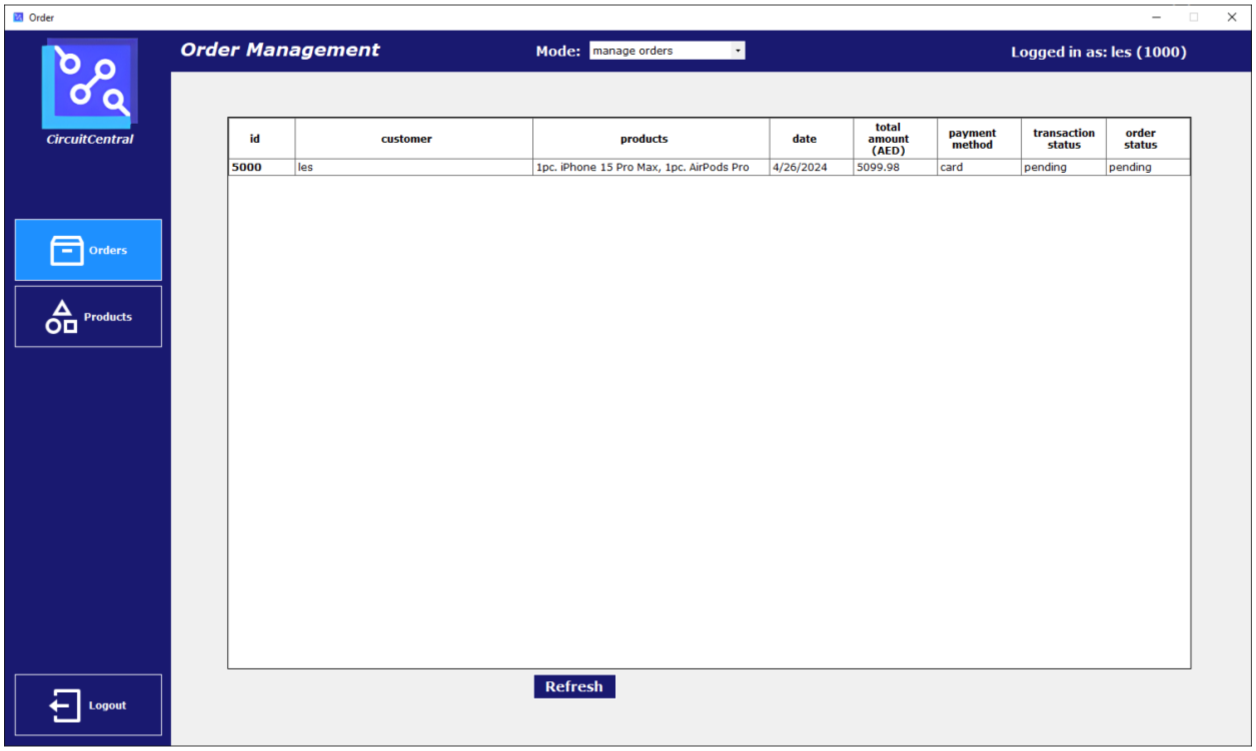
id	name	unit price (AED)	quantity
3001	iPhone 15 Pro Max	4499.99	1
3002	AirPods Pro	599.99	1

SUCCESS: order placed!

Figure 55: Adding product to order as user

In the example above, Product 3001 is added to the cart (right table). The user can add as many products, increasing the quantity of the item. Afterwards, a new order record is created. This is done by entering the necessary details on the text fields and adding products to the cart (right table).

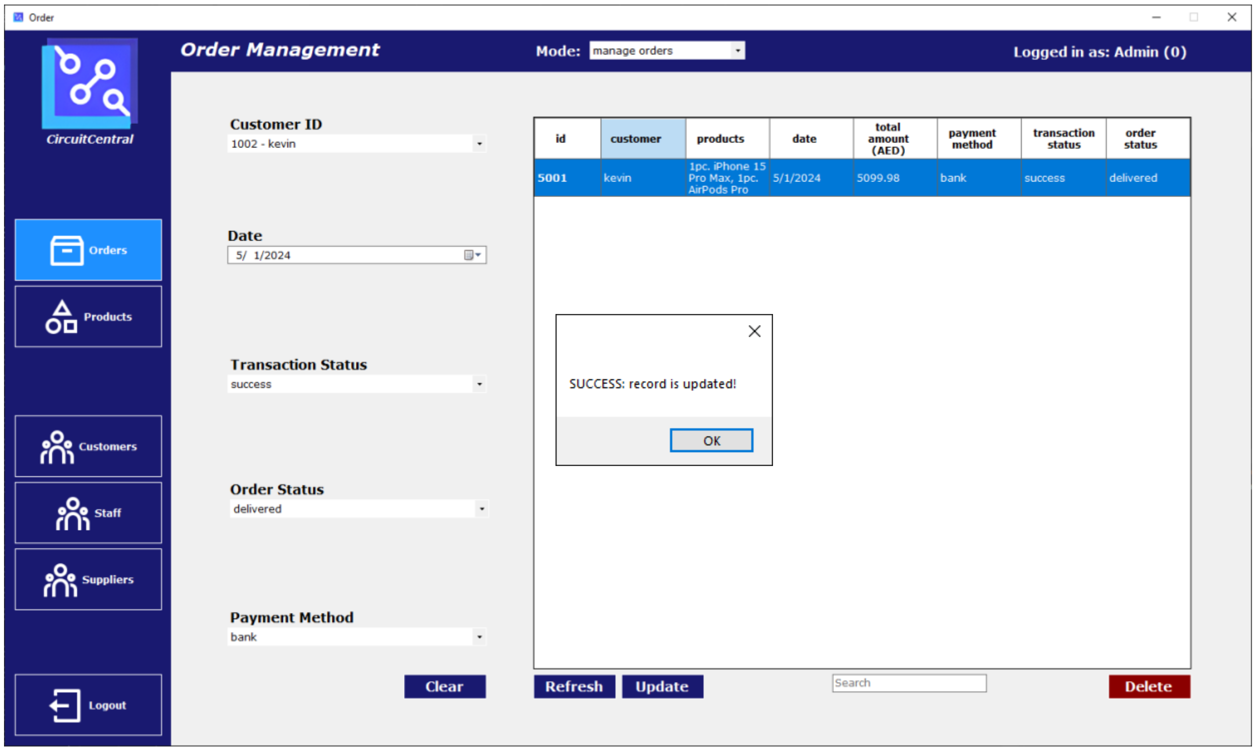
Note that items in the cart can be deleted by selecting the item in the cart, and then clicking the delete button.



id	customer	products	date	total amount (AED)	payment method	transaction status	order status
5000	les	1pc. iPhone 15 Pro Max, 1pc. AirPods Pro	4/26/2024	5099.98	card	pending	pending

Figure 56: Viewing order as user

In this view, users can ONLY see their own orders. This is used to track order details regarding their order. Users are unable to edit ANY details, however.



Order

Order Management Mode: manage orders Logged in as: Admin (0)

CircuitCentral

Customer ID: 1002 - kevin

Date: 5/ 1/2024

Transaction Status: success

Order Status: delivered

Payment Method: bank

id	customer	products	date	total amount (AED)	payment method	transaction status	order status
5001	kevin	1pc. iPhone 15 Pro Max, 1pc. AirPods Pro	5/1/2024	5099.98	bank	success	delivered

SUCCESS: record is updated!

Figure 57: Updating order record as admin

Using the previous example, we updated Order 5001, which previously has a transaction status and order status as pending, to transaction status being success and order status being delivered. This is done by highlighting a row, then entering the updated details in the text fields, and clicking the update button.

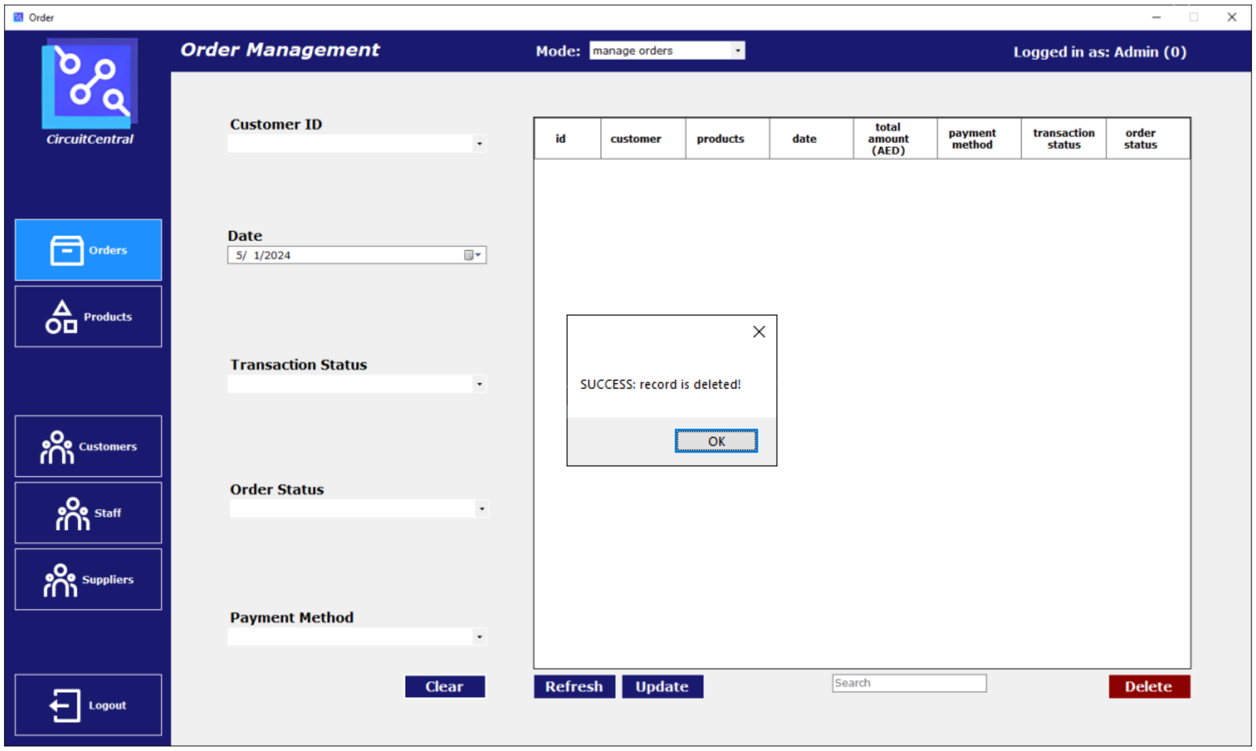


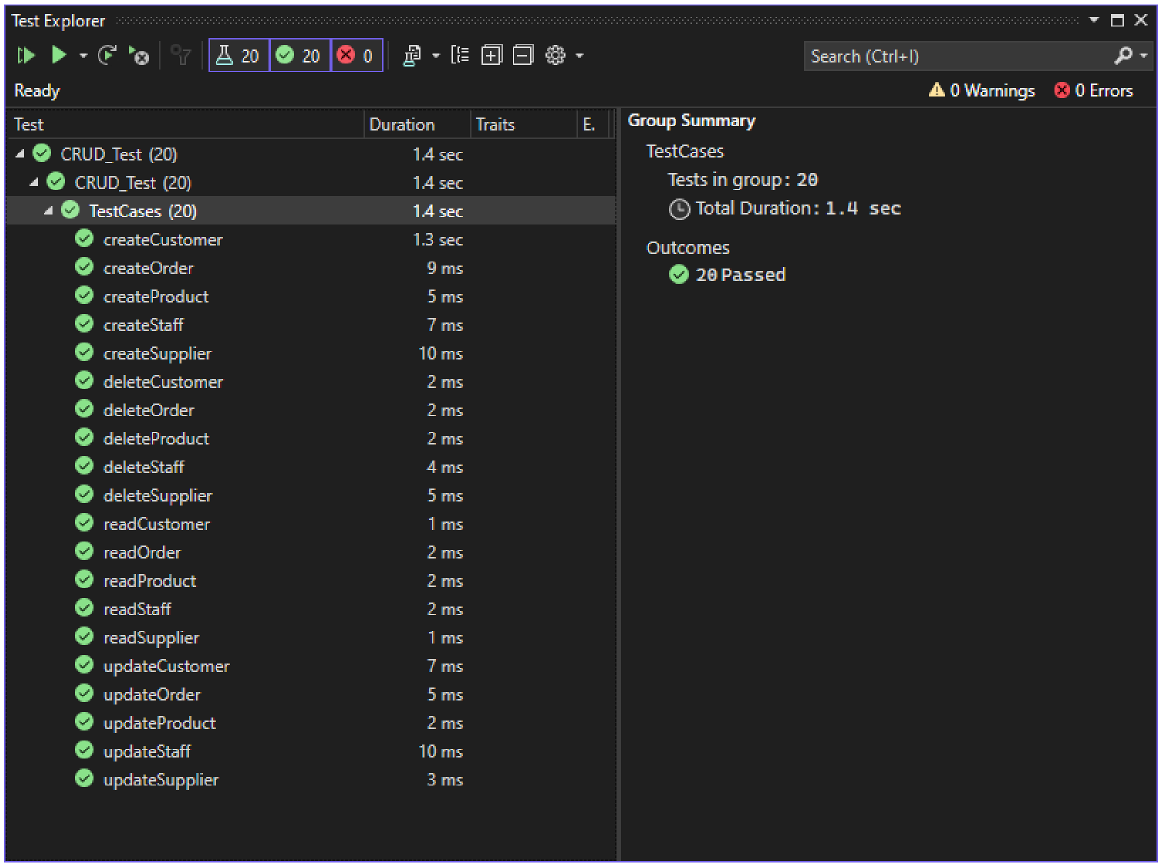
Figure 58: Deleting order record

Using the previous example, we deleted Order 5001. This is done by highlighting a row and then clicking the delete button.

TEST-DRIVEN DEVELOPMENT (TEST CASES)

Test-drive Development (TDD) is a software development approach where codes are written before officially writing codes on a much bigger codebase **(Mikejo5000, 2023)**. This is especially useful when refactoring code as developers can repeat this cycle for improving and developing new code functionalities.

Visual Studio 2022 was used to test each CRUD operation of each Entity (Customer, Staff, etc.). The methods were extracted from the main project and imported inside the test environment. With a total of 20 methods, each of them passed with a green check mark. Below is the screenshot of TDD.



Test	Duration	Traits	E.
CRUD_Test (20)	1.4 sec		
CRUD_Test (20)	1.4 sec		
TestCases (20)	1.4 sec		
createCustomer	1.3 sec		
createOrder	9 ms		
createProduct	5 ms		
createStaff	7 ms		
createSupplier	10 ms		
deleteCustomer	2 ms		
deleteOrder	2 ms		
deleteProduct	2 ms		
deleteStaff	4 ms		
deleteSupplier	5 ms		
readCustomer	1 ms		
readOrder	2 ms		
readProduct	2 ms		
readStaff	2 ms		
readSupplier	1 ms		
updateCustomer	7 ms		
updateOrder	5 ms		
updateProduct	2 ms		
updateStaff	10 ms		
updateSupplier	3 ms		

Group Summary
TestCases
Tests in group: 20
Total Duration: 1.4 sec
Outcomes
20 Passed

Figure 59: Test cases in Visual Studio 2022

CUSTOMER

```
[TestMethod]
public void createCustomer()
{
    CustomerDetails customer = new CustomerDetails();
    customer.name = "Daniel Martinez";
    customer.email_address = "daniel.martinez@email.com";
    customer.password = "021504";
    customer.phone_number = "+1 (555) 777-8888";
    customer.street = "606 Oak Street";
    customer.city = "Someville";
    customer.postal_code = "6789";
    customer.country = "Australia";

    crud.createCustomer(customer);
}
```

Figure 60: createCustomer() test case

```
[TestMethod]
public void readCustomer()
{
    crud.readCustomer(1000);
}
```

Figure 61: readCustomer() test case

```
[TestMethod]
public void updateCustomer()
{
    CustomerDetails customer = new CustomerDetails();
    customer.name = "Kevin Nguyen";
    customer.email_address = "kevin.nguyen@email.com";
    customer.password = "6789";
    customer.phone_number = "+1 (555) 333-4444";
    customer.street = "404 Cedar Avenue";
    customer.city = "Othertown";
    customer.postal_code = "75843";
    customer.country = "Canada";



    crud.updateCustomer(1000, 1, customer);
}
```

Figure 62: updateCustomer() test case

```
[TestMethod]
public void deleteCustomer()
{
    crud.deleteCustomer(1000, 1);
}
```

Figure 63: deleteCustomer() test case

STAFF

```
[TestMethod]
public void createStaff()
{
    StaffDetails staff = new StaffDetails();
    staff.name = "John Smith";
    staff.email_address = "john.smith@email.com";
    staff.password = "12345";
    staff.phone_number = "+1 (555) 123-4567";
    staff.department = "sales & marketing";
    staff.street = "123 Main Street";
    staff.city = "Anytown";
    staff.postal_code = "12345";
    staff.country = "United Arab Emirates";

    crud.createStaff(staff);
}
```

Figure 64: createStaff() test case

```
[TestMethod]
public void readStaff()
{
    crud.readStaff(500);
}
```

Figure 65: readStaff() test case

```
[TestMethod]
public void updateStaff()
{
    StaffDetails staff = new StaffDetails();
    staff.name = "Emily Johnson";
    staff.email_address = "emily.johnson@email.com";
    staff.password = "54321";
    staff.phone_number = "+1 (555) 987-6543";
    staff.department = "technology & it";
    staff.street = "456 Elm Street";
    staff.city = "Othertown";
    staff.postal_code = "54321";
    staff.country = "United Arab Emirates";

    crud.updateStaff(500, 4, staff);
}
```

Figure 66: updateStaff() test case

```
[TestMethod]
public void deleteStaff()
{
    crud.deleteStaff(500, 4);
}
```

Figure 67: deleteStaff() test case

SUPPLIER

```
[TestMethod]
public void createSupplier()
{
    SupplierDetails supplier = new SupplierDetails();
    supplier.name = "ABC Corporation";
    supplier.email_address = "abc@email.com";
    supplier.phone_number = "+1 (555) 123-4567";
    supplier.street = "123 Main Street";
    supplier.city = "Anytown";
    supplier.postal_code = "12345";
    supplier.country = "United States";

    crud.createSupplier(supplier);
}
```

Figure 68: createSupplier() test case

```
[TestMethod]
public void readSupplier()
{
    crud.readSupplier(7000);
}
```

Figure 69: readSupplier() test case

```
[TestMethod]
public void updateSupplier()
{
    SupplierDetails supplier = new SupplierDetails();
    supplier.name = "XYZ Enterprises";
    supplier.email_address = "xyz@email.com";
    supplier.phone_number = "+1 (555) 987-6543";
    supplier.street = "456 Elm Street";
    supplier.city = "Othertown";
    supplier.postal_code = "54321";
    supplier.country = "Canada";
    crud.updateSupplier(7000, 6, supplier);
}
```

Figure 70: updateSupplier() test case

```
[TestMethod]
public void deleteSupplier()
{
    crud.deleteSupplier(7000, 6);
}
```

Figure 71: deleteSupplier() test case

PRODUCT

```
[TestMethod]
public void createProduct()
{
    ProductDetails product = new ProductDetails();
    product.name = "SONY WH-1000XM4";
    product.description = "Headphones";
    product.category = "audio & headphones";
    product.price = 799.99M;
    product.stock = 46;
    product.image_link = "C:\\Users\\les\\Downloads\\LANDrop\\41xvg7mwu3L._AC_SL1000_.jpg";

    crud.createProduct(7001, product);
}
```

Figure 72: createProduct() test case

```
[TestMethod]
public void readProduct()
{
    crud.readProduct(3000);
}
```

Figure 73: readProduct() test case

```
[TestMethod]
public void updateProduct()
{
    ProductDetails product = new ProductDetails();

    product.name = "Bose QuietComfort Ultra";
    product.description = "Headphones";
    product.category = "audio & headphones";
    product.price = 1699.99M;
    product.stock = 16;
    product.image_link = "C:\\Users\\les\\Downloads\\LANDrop\\51NC9ErIQtL._AC_SL1500_.jpg";

    crud.updateProduct(3000, 7002, product);
}
```

Figure 74: updateProduct() test case

```
[TestMethod]
public void deleteProduct()
{
    crud.deleteProduct(3000);
}
```

Figure 75: deleteProduct() test case

ORDER

```
[TestMethod]
public void createOrder()
{
    crud.createOrder(1001, 3001, "card");
}
```

Figure 76: createOrder() test case

```
[TestMethod]
public void readOrder()
{
    crud.readOrder(1001);
}
```

Figure 77: readOrder() test case

```
[TestMethod]
public void updateOrder()
{
    DateTime date = new DateTime(2024, 4, 30);
    crud.updateOrder(5000, 9000, 1002, date, "success", "COMPLETED", "bank");
}
```

Figure 78: updateOrder() test case

```
[TestMethod]
public void deleteOrder()
{
    crud.deleteOrder(5000, 9000);
}
```

Figure 79: deleteOrder() test case

CONCLUSION

This report has delved into the development of an e-commerce management system made in Windows Forms, together with C#. CircuitCentral showcases versatility and robustness of the C# programming language. With that, the report has covered various concepts of the project, such as OOPs, WinForms, Visual Studio, refactorization, GitHub, etc.

In the methodology section, the report has explained every detail on why each component was chosen and how it benefits the project. Along with that, screenshots and explanations of code and UI designs were provided for easier understanding.

REFERENCES

adegeo (2023). *What is Windows Forms - Windows Forms .NET*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-8.0.

Doherty, E. (2020). *What is Object Oriented Programming? OOP Explained in Depth*. [online] Educative: Interactive Courses for Software Developers. Available at: https://www.educative.io/blog/object-oriented-programming.

Hughes, A. (2019). *What is Microsoft SQL Server? A definition from WhatIs.com*. [online] SearchDataManagement. Available at: https://www.techtarget.com/searchdatamanagement/definition/SQL-Server.

Jalli, A. (2022). *What Is C#? (Definition, Uses, Syntax, Tools)* | *Built In*. [online] builtin.com. Available at: https://builtin.com/software-engineering-perspectives/c-sharp.

Kara, K. (2023). *The Art of Code Refactoring in C#*. [online] Medium. Available at: https://medium.com/@kerimkkara/the-art-of-code-refactoring-in-c-d02f0346a1dd [Accessed 25 Apr. 2024].

Korbin Brown (2016). *What Is GitHub, and What Is It Used For?* [online] How-To Geek. Available at: https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/.

Microsoft (2023). *Overview of Visual Studio*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022.

Mikejo5000 (2023). *Test-driven development walkthrough - Visual Studio (Windows)*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/visualstudio/test/quick-start-test-driven-development-with-test-explorer?view=vs-2022.