NATIONAL POST GRADUATE COLLEGE



**INVENTORY MANAGEMENT SYSTEM (C#)**

Submitted by: PRIYANSHU BATHAM

Class: BCA-5

ID No: 722017

Submitted to: MR.AMIT SRIVASTAVA

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **TOPIC** | **Pg. No** | **Date** | **Sign** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**DATA MODELS:**

/\*This library contains all the Db-Tables equivalent classes

\* so that the data returned by SQL queries can be managed easily

\* in the codebase.

\*/

using DataAccessLayer;

namespace DataModels

{

public class Captcha

{

public string id;

public string Text;

public string Path;

public Captcha(string id, string text, string path)

{

this.id = id;

this.Text = text;

this.Path = path;

}

}

public class ShippingAddress{

public string? id { get; set; }

public string customer\_id { get; set; }

public string city { get; set; }

public string state { get; set; }

public string country { get; set; }

public string? more { get; set; }

public ShippingAddress(string customer\_id, string city, string state, string country, string more = "") {

this.customer\_id = customer\_id;

this.city = city;

this.state = state;

this.country = country;

this.more = more;

}

override

public string ToString()

{

return $"{id}) {city}, {state}, {country}";

}

}

public class LoginSession {

public string? id { get; set; }

public string employee\_id { get; set; }

public string? loggedInAt { get; set; }

public LoginSession(string employee\_id)

{

this.employee\_id = employee\_id;

}

}

public class Employee

{

public string? id { get; set; }

public decimal salary { get; set; }

public Role role { get; set; }

public string passwordHash { get; set; }

public Employee(decimal salary, Role role, string passwordHash)

{

this.salary = salary;

this.role = role;

this.passwordHash = passwordHash;

}

override

public string ToString()

{

return $"{id}) {role}";

}

}

public class Supplier

{

public string? id { get; set; }

public string email { get; set; }

public Supplier\_Type supplier\_type { get; set; }

public Supplier(string email, Supplier\_Type supplier\_type)

{

this.email = email;

this.supplier\_type = supplier\_type;

}

override

public string ToString()

{

User user = SqlUser.getBySupplierId(this.id);

return $"{id}) {user.firstName}";

}

}

public class Customer

{

public string? id { get; set; }

public string? supplier { get; set; }

public string email { get; set; }

public Customer\_Type customer\_type { get; set; }

public Customer(string email, Customer\_Type customer\_type)

{

this.email = email;

this.customer\_type = customer\_type;

}

override

public string ToString()

{

return $"{id}) {email}";

}

}

public class Brand

{

public string? id { get; set; }

public string title { get; set; }

public string? summary { get; set; }

public Popularity popularity { get; set; }

public Brand(string title, Popularity popularity, string? summary = null)

{

this.title = title;

this.summary = summary;

this.popularity = popularity;

}

override

public string ToString()

{

return $"{id}) {title}";

}

}

public class Payment

{

public string? id { get; set; }

public string user\_id { get; set; }

public string order\_id { get; set; }

public string shippingAddress\_id{ get; set; }

public Mode mode{ get; set; }

public Status status { get; set; }

public string? createdAt { get; set; }

public Type type { get; set; }

public Payment(string user\_id, string order\_id, string shippingAddress\_id, Mode mode, Status status, Type type, string? createdAt = "")

{

this.user\_id = user\_id;

this.order\_id = order\_id;

this.shippingAddress\_id = shippingAddress\_id;

this.mode = mode;

this.status = status;

this.createdAt = createdAt;

this.type = type;

}

}

public class Order\_Item

{

public string? id { get; set; }

public string product\_id { get; set; }

public string item\_id { get; set; }

public string order\_id { get; set; }

public decimal price { get; set; }

public long quantity { get; set; }

public decimal total\_price { get; set; }

public Order\_Item(string product\_id, string item\_id, string order\_id, decimal price, long quantity, decimal total\_price)

{

this.product\_id = product\_id;

this.item\_id = item\_id;

this.order\_id = order\_id;

this.price = price;

this.quantity = quantity;

this.total\_price = total\_price;

}

}

public class Item

{

public string? id { get; set; }

public string product\_id { get; set; }

public string brand\_id { get; set; }

public string supplier\_id { get; set; }

public decimal price { get; set; }

public int discount { get; set; }

public long quantity { get; set; }

public decimal stockValue { get; set; }

public long alarm\_quantity{ get; set; }

public Item(string product\_id, string brand\_id, string supplier\_id, decimal price, int discount, long quantity, decimal stockValue, long alarm\_quantity) {

this.product\_id= product\_id;

this.brand\_id= brand\_id;

this.supplier\_id= supplier\_id;

this.price = price;

this.discount = discount;

this.quantity = quantity;

this.stockValue = stockValue;

this.alarm\_quantity = alarm\_quantity;

}

}

public class Order

{

public string? id { get; set; }

public string user\_id { get; set; }

public string employee\_id { get; set; }

public Type type { get; set; }

public decimal subTotal { get; set; }

public decimal tax { get; set; }

public decimal total { get; set; }

public Order(string user\_id, string employee\_id, Type type, decimal subTotal, decimal tax, decimal total)

{

this.user\_id = user\_id;

this.employee\_id = employee\_id;

this.type = type;

this.subTotal = subTotal;

this.tax = tax;

this.total = total;

}

}

public class User

{

public string? id { get; set; }

public string? supplier\_id { get; set; }

public string? customer\_id { get; set; }

public string? employee\_id { get; set; }

public User\_Type user\_type { get; set; }

public string firstName { get; set; }

public string? lastName{ get; set; }

public string username{ get; set; }

public string mobile { get; set; }

public string email { get; set; }

public string address { get; set; }

public string? registeredAt { get; set; }

public User(User\_Type user\_type, string firstName, string username, string mobile, string email, string address, string? supplier\_id = null, string? customer\_id = null, string? employee\_id = null, string lastName = "", string registeredAt = "")

{

this.supplier\_id = supplier\_id;

this.customer\_id = customer\_id;

this.employee\_id = employee\_id;

this.user\_type = user\_type;

this.firstName = firstName;

this.lastName = lastName;

this.username = username;

this.mobile = mobile;

this.email = email;

this.address = address;

this.registeredAt = registeredAt;

}

}

public class Product\_Category

{

public string? id { get; set; }

public string category\_id { get; set; }

public string product\_id { get; set; }

public Product\_Category(string category\_id, string product\_id)

{

this.category\_id = category\_id;

this.product\_id = product\_id;

}

}

public class Category

{

public string? id { get; set; }

public string title { get; set; }

public string? description { get; set; }

public Category(string title, string? description = null)

{

this.title = title;

this.description = description;

}

}

public class Product

{

public string? id { get; set; }

public string title { get; set; }

public string? description { get; set; }

public string? createdAt { get; set; }

public Product(string title, string? description = null)

{

this.title = title;

this.description = description;

}

override

public string ToString()

{

return $"{id}) {title}";

}

}

//Enumerations for Consistency

public enum Role

{

Manager,

Sales

}

public enum Supplier\_Type

{

Trusted,

New

}

public enum Customer\_Type

{

Rich,

Poor,

Medium

}

public enum Popularity {

Low,

Medium,

High

}

public enum Mode

{

Online,

Cod

}

public enum Status

{

Pending,

Finished,

Failed

}

public enum Type

{

In,

Out

}

public enum User\_Type

{

Customer,

Supplier,

Employee

}

}

**DATA ACCESS LAYER:**

using DataModels;

using Microsoft.Data.SqlClient;

using System.Collections.Generic;

namespace DataAccessLayer

{

//Parent class for holding the connection object

public class SqlHelper

{

//private as there's no need to access it from anywhere

private static string \_connString = "Data Source=PRIYANSHU\\SQLEXPRESS;Initial Catalog=Inventory\_Management\_System;Integrated Security=True;TrustServerCertificate=True";

//protected coz i'll use conn in the child classes

protected static SqlConnection conn = new SqlConnection(\_connString);

}

//Captcha---------------------->>>>

public class SqlCaptcha : SqlHelper {

public static int fetchNoOfCaptchas()

{

try

{

string query = "select COUNT(\*) from captcha;";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

int count = Convert.ToInt32(cmd.ExecuteScalar());

conn.Close();

cmd.Dispose();

return count;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

public static Captcha fetchRandomCaptcha(int noOfCaptchas)

{

try

{

Random random = new Random();

int id = random.Next(noOfCaptchas) + 1; //+1 coz I don't want zero index;

string query = $"select \* from captcha where id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

string text = reader["text"].ToString()!;

string path = reader["path"].ToString()!;

Captcha captcha = new Captcha(id.ToString(), text, path);

conn.Close();

cmd.Dispose();

return captcha;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//ShippingAddress----------------->>>>

public class SqlShippingAddress : SqlHelper

{

//returns all shipping addresses of a customer

public static List<ShippingAddress> getMany(string customer\_id)

{

try

{

List<ShippingAddress> list = new List<ShippingAddress>();

string query = $"select \* from shippingAddress where customer\_id = {customer\_id}";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

ShippingAddress shippingAddress = new ShippingAddress(reader["customer\_id"].ToString()!, reader["city"].ToString()!, reader["state"].ToString()!, reader["country"].ToString()!, reader["more"].ToString()!);

shippingAddress.id = reader["id"].ToString();

list.Add(shippingAddress);

}

reader.Close();

cmd.Dispose();

conn.Close();

return list;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//getById

public static ShippingAddress getById(string id)

{

try

{

string query = $"select \* from shippingAddress where id = {id}";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

ShippingAddress shippingAddress = new ShippingAddress(reader["customer\_id"].ToString()!, reader["city"].ToString()!, reader["state"].ToString()!, reader["country"].ToString()!, reader["more"].ToString()!);

shippingAddress.id = reader["id"].ToString();

reader.Close();

cmd.Dispose();

conn.Close();

return shippingAddress;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//adds a new shippingAddress of a customer and returns shippingAddress object with the id

public static ShippingAddress add(ShippingAddress shippingAddress)

{

try

{

string query = $"insert into shippingAddress (customer\_id, city, state, country, more) values (@customer\_id, @city, @state, @country, @more); select SCOPE\_IDENTITY()";

SqlCommand cmd = new SqlCommand(query, conn);

//if the members of object are null then equivalent Db null is added in Sql

cmd.Parameters.AddWithValue("@customer\_id", shippingAddress.customer\_id);

cmd.Parameters.AddWithValue("@city", shippingAddress.city);

cmd.Parameters.AddWithValue("@state", shippingAddress.state);

cmd.Parameters.AddWithValue("@country", shippingAddress.country);

cmd.Parameters.AddWithValue("@more", shippingAddress.more ?? (object)DBNull.Value);

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

shippingAddress.id = id;

cmd.Dispose();

conn.Close();

return shippingAddress;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//delete

public static void delete(string id)

{

try

{

string query = $"delete from shippingAddress where id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

cmd.ExecuteNonQuery();

cmd.Dispose();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//update

public static void update(ShippingAddress shippingAddress)

{

try

{

string query = $"update shippingAddress set city = @city, state = @state, country = @country, more = @more where id = @id;";

SqlCommand cmd = new SqlCommand(query, conn);

// Adding parameters with null checks

cmd.Parameters.AddWithValue("@city", shippingAddress.city);

cmd.Parameters.AddWithValue("@state", shippingAddress.state);

cmd.Parameters.AddWithValue("@country", shippingAddress.country);

cmd.Parameters.AddWithValue("@more", shippingAddress.more ?? (object)DBNull.Value);

cmd.Parameters.AddWithValue("@id", shippingAddress.id);

conn.Open();

cmd.ExecuteNonQuery();

cmd.Dispose();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//getAll

public static List<ShippingAddress> getAll()

{

try

{

List<ShippingAddress> list = new List<ShippingAddress>();

string query = $"select \* from shippingAddress;";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

ShippingAddress shippingAddress = new ShippingAddress(reader["customer\_id"].ToString()!, reader["city"].ToString()!, reader["state"].ToString()!, reader["country"].ToString()!, reader["more"].ToString()!);

shippingAddress.id = reader["id"].ToString();

list.Add(shippingAddress);

}

reader.Close();

cmd.Dispose();

conn.Close();

return list;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//makeSelf

public static void makeSelf()

{

try

{

string query = $"insert into shippingAddress (customer\_id, city, state, country, more) values (@customer\_id, @city, @state, @country, @more); select SCOPE\_IDENTITY()";

SqlCommand cmd = new SqlCommand(query, conn);

//if the members of object are null then equivalent Db null is added in Sql

cmd.Parameters.AddWithValue("@customer\_id", SqlCustomer.getSelfId());

cmd.Parameters.AddWithValue("@city", "self");

cmd.Parameters.AddWithValue("@state", "self");

cmd.Parameters.AddWithValue("@country", "self");

cmd.Parameters.AddWithValue("@more", "self" ?? (object)DBNull.Value);

conn.Open();

cmd.ExecuteNonQuery();

cmd.Dispose();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

public static string getSelfId()

{

try

{

string query = $"select id from shippingAddress where customer\_id = @customer\_id;";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@customer\_id", SqlCustomer.getSelfId());

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

string id = reader["id"].ToString()!;

cmd.Dispose();

conn.Close();

return id;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//LoginSession----------------->>>>

public class SqlLoginSession: SqlHelper

{

//returns all Login Sessions of an employee

public static List<LoginSession> getMany(Employee employee)

{

try

{

List<LoginSession> list = new List<LoginSession>();

string query = $"select \* from loginSession where employee\_id = {employee.id}";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

LoginSession loginSession = new LoginSession(employee.id!);

loginSession.id = reader["id"].ToString();

loginSession.loggedInAt = reader["loggedInAt"].ToString();

list.Add(loginSession);

}

reader.Close();

cmd.Dispose();

conn.Close();

return list;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//creates a loginSession for an employee and returns loginSession object with id and loggedInAt values

public static LoginSession add(Employee employee)

{

try

{

LoginSession loginSession = new LoginSession(employee.id!);

string query = $"insert into loginSession (employee\_id) values (@employee\_id); select SCOPE\_IDENTITY()";

SqlCommand cmd = new SqlCommand(query, conn);

//if the members of object are null then equivalent Db null is added in Sql

cmd.Parameters.AddWithValue("@employee\_id", employee.id);

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

loginSession.id = id;

cmd.Dispose();

//this additional part is for getting the loggedInAt value generated automatically in sql

query = $"select loggedInAt from loginSession where id = {loginSession.id}";

cmd = new SqlCommand(query, conn);

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

loginSession.loggedInAt = reader["loggedInAt"].ToString()!;

reader.Close();

cmd.Dispose();

conn.Close();

return loginSession;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//Employee----------------->>>>

public class SqlEmployee: SqlHelper

{

//adds a new Employee and returns Employee object with the id

public static Employee add(Employee employee)

{

try

{

string query = $"insert into employee (salary, role, passwordHash) values (@salary, @role, @passwordHash); select SCOPE\_IDENTITY();";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@salary", employee.salary);

cmd.Parameters.AddWithValue("@role", employee.role.ToString());

cmd.Parameters.AddWithValue("@passwordHash", employee.passwordHash);

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

employee.id = id;

cmd.Dispose();

conn.Close();

return employee;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//getAll

public static List<Employee> getAll()

{

try

{

List<Employee> employees = new List<Employee>();

string query = $"select \* from employee;";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

Employee employee = new Employee(Convert.ToDecimal(reader["salary"].ToString()), (Role)Enum.Parse(typeof(Role), reader["role"].ToString()!), reader["passwordHash"].ToString()!);

employee.id = reader["id"].ToString()!;

employees.Add(employee);

}

cmd.Dispose();

conn.Close();

return employees;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//getById

public static Employee getById(string id)

{

try

{

string query = $"select \* from employee where id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

Employee employee = new Employee(Convert.ToDecimal(reader["salary"].ToString()), (Role)Enum.Parse(typeof(Role), reader["role"].ToString()!), reader["passwordHash"].ToString()!);

employee.id = reader["id"].ToString()!;

cmd.Dispose();

conn.Close();

return employee;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//delete

public static void delete(string id)

{

try

{

string query = $"delete from employee where id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//update

public static void update(Employee employee)

{

try

{

string query = $"update employee set salary = @salary, role = @role, passwordHash = @passwordHash where id = @id;";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@id", employee.id);

cmd.Parameters.AddWithValue("@salary", employee.salary);

cmd.Parameters.AddWithValue("@role", employee.role.ToString());

cmd.Parameters.AddWithValue("@passwordHash", employee.passwordHash);

conn.Open();

cmd.ExecuteNonQuery();

cmd.Dispose();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//Supplier----------------->>>>

public class SqlSupplier: SqlHelper

{

//adds a new Supplier and returns Supplier object with the id

public static Supplier add(Supplier supplier)

{

try

{

string query = $"insert into supplier (email, supplier\_type) values (@email, @supplier\_type); select SCOPE\_IDENTITY();";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@email", supplier.email);

cmd.Parameters.AddWithValue("@supplier\_type", supplier.supplier\_type.ToString());

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

supplier.id = id;

cmd.Dispose();

conn.Close();

return supplier;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

public static List<Supplier> getAll()

{

try

{

List<Supplier> list = new List<Supplier>();

string query = $"select \* from supplier;";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

Supplier supplier = new Supplier(reader["email"].ToString()!, (Supplier\_Type)Enum.Parse(typeof(Supplier\_Type), reader["supplier\_type"].ToString()!));

supplier.id = reader["id"].ToString()!;

list.Add(supplier);

}

conn.Close();

return list;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//getById

public static Supplier getById(string id)

{

try

{

string query = $"select \* from supplier where id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

Supplier supplier = new Supplier(reader["email"].ToString()!, (Supplier\_Type)Enum.Parse(typeof(Supplier\_Type), reader["supplier\_type"].ToString()!));

supplier.id = reader["id"].ToString()!;

cmd.Dispose();

conn.Close();

return supplier;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//delete

public static void delete(string id)

{

try

{

string query = $"delete from supplier where id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//update

public static void update(Supplier supplier)

{

try

{

string query = $"update supplier set email = @email, supplier\_type = @supplier\_type where id = @id;";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@id", supplier.id);

cmd.Parameters.AddWithValue("@email", supplier.email);

cmd.Parameters.AddWithValue("@customer\_type", supplier.supplier\_type.ToString());

conn.Open();

cmd.ExecuteNonQuery();

cmd.Dispose();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//Customer----------------->>>>

public class SqlCustomer : SqlHelper

{

//adds a new Customer and returns Customer object with the id

public static Customer add(Customer customer)

{

try

{

string query = $"insert into customer (email, customer\_type) values (@email, @customer\_type); select SCOPE\_IDENTITY();";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@email", customer.email);

cmd.Parameters.AddWithValue("@customer\_type", customer.customer\_type.ToString());

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

customer.id = id;

cmd.Dispose();

conn.Close();

return customer;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//getAll

public static List<Customer> getAll()

{

try

{

List<Customer> customers = new List<Customer>();

string query = $"select \* from customer;";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

Customer customer = new Customer(reader["email"].ToString()!, (Customer\_Type)Enum.Parse(typeof(Customer\_Type), reader["customer\_type"].ToString()!));

customer.id = reader["id"].ToString()!;

customers.Add(customer);

}

cmd.Dispose();

conn.Close();

return customers;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//getById

public static Customer getById(string id)

{

try

{

string query = $"select \* from customer where id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

Customer customer = new Customer(reader["email"].ToString()!, (Customer\_Type)Enum.Parse(typeof(Customer\_Type), reader["customer\_type"].ToString()!));

customer.id = reader["id"].ToString()!;

cmd.Dispose();

conn.Close();

return customer;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//delete

public static void delete(string id)

{

try

{

string query = $"delete from customer where id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//update

public static void update(Customer customer)

{

try

{

string query = $"update customer set email = @email, customer\_type = @customer\_type where id = @id;";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@id", customer.id);

cmd.Parameters.AddWithValue("@email", customer.email);

cmd.Parameters.AddWithValue("@customer\_type", customer.customer\_type.ToString());

conn.Open();

cmd.ExecuteNonQuery();

cmd.Dispose();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

// make self customer

public static void makeself()

{

try

{

string query = $"insert into customer (email, customer\_type) values (@email, @customer\_type);";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@email", "self@gmail.com");

cmd.Parameters.AddWithValue("@customer\_type", Customer\_Type.Rich.ToString());

conn.Open();

cmd.ExecuteNonQuery();

cmd.Dispose();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

public static string getSelfId()

{

try

{

string query = $"select id from customer where email = 'self@gmail.com';";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

string id = reader["id"].ToString()!;

cmd.Dispose();

conn.Close();

return id;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//Brand----------------->>>>

public class SqlBrand: SqlHelper

{

//adds a new Brand and returns Brand object with the id

public static Brand add(Brand brand)

{

try

{

string query = $"insert into brand (title, summary, popularity) values (@title, @summary, @popularity); select SCOPE\_IDENTITY();";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@title", brand.title);

cmd.Parameters.AddWithValue("@summary", brand.summary ?? (object)DBNull.Value);

cmd.Parameters.AddWithValue("@popularity", brand.popularity.ToString());

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

brand.id = id;

cmd.Dispose();

conn.Close();

return brand;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

public static List<Brand> getAll()

{

try

{

List<Brand> list = new List<Brand>();

string query = $"select \* from brand;";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

Brand brand = new Brand(reader["title"].ToString()!, (Popularity)Enum.Parse(typeof(Popularity), reader["popularity"].ToString()!), reader["summary"].ToString());

brand.id = reader["id"].ToString()!;

list.Add(brand);

}

conn.Close();

return list;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//User----------------->>>>

public class SqlUser : SqlHelper

{

//adds a new User and returns User object with the id

//1)remember to pass the appropriate employee OR customer OR supplier id. One of these is mandatory !!!!

//2) firstly an employee, customer OR supplier will be added then using its id this user will be created

public static User add(User user)

{

//user is reserved word in sqlserver so we have to write it in [user] like this

string query = $"insert into [user] (user\_type, firstName, lastName, username, mobile, email, address, supplier\_id, customer\_id, employee\_id) values (@user\_type, @firstName, @lastName, @username, @mobile, @email, @address, @supplier\_id, @customer\_id, @employee\_id); select SCOPE\_IDENTITY();";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@user\_type", user.user\_type.ToString());

cmd.Parameters.AddWithValue("@firstName", user.firstName);

cmd.Parameters.AddWithValue("@lastName", user.lastName ?? (object)DBNull.Value);

cmd.Parameters.AddWithValue("@username", user.username);

cmd.Parameters.AddWithValue("@mobile", user.mobile);

cmd.Parameters.AddWithValue("@email", user.email);

cmd.Parameters.AddWithValue("@address", user.address);

cmd.Parameters.AddWithValue("@supplier\_id", user.supplier\_id ?? (object)DBNull.Value);

cmd.Parameters.AddWithValue("@customer\_id", user.customer\_id ?? (object)DBNull.Value);

cmd.Parameters.AddWithValue("@employee\_id", user.employee\_id ?? (object)DBNull.Value);

try

{

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

user.id = id;

cmd.Dispose();

//this extra work to refetch the record we added and pull the registeredAt field

query = $"select registeredAt from [user] where id = {id}";

cmd = new SqlCommand(query, conn);

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

user.registeredAt = reader["registeredAt"].ToString()!;

reader.Close();

conn.Close();

return user;

}

catch (Exception ex)

{

//also now delete the supplier/customer/employee created before this fun was called

conn.Close();

if(user.customer\_id != null)

{

SqlCustomer.delete(user.customer\_id);

}

else if(user.supplier\_id != null)

{

SqlSupplier.delete(user.supplier\_id);

}

else if (user.employee\_id!= null)

{

SqlSupplier.delete(user.employee\_id);

}

throw new Exception(ex.Message);

}

}

//get user by employee id

public static User getByEmployeeId(string id)

{

try

{

string query = $"select \* from [user] where employee\_id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

User user = new User(User\_Type.Employee, reader["firstName"].ToString()!, reader["username"].ToString()!, reader["mobile"].ToString()!, reader["email"].ToString()!, reader["address"].ToString()!, employee\_id: id, lastName: reader["lastName"].ToString()!, registeredAt: reader["registeredAt"].ToString()!);

user.id = reader["id"].ToString()!;

conn.Close();

return user;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//get user by customer id

public static User getByCustomerId(string id)

{

try

{

string query = $"select \* from [user] where customer\_id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

User user = new User(User\_Type.Customer, reader["firstName"].ToString()!, reader["username"].ToString()!, reader["mobile"].ToString()!, reader["email"].ToString()!, reader["address"].ToString()!, employee\_id: id, lastName: reader["lastName"].ToString()!, registeredAt: reader["registeredAt"].ToString()!);

user.id = reader["id"].ToString()!;

conn.Close();

return user;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//get user by supplier id

public static User getBySupplierId(string id)

{

try

{

string query = $"select \* from [user] where supplier\_id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

User user = new User(User\_Type.Supplier, reader["firstName"].ToString()!, reader["username"].ToString()!, reader["mobile"].ToString()!, reader["email"].ToString()!, reader["address"].ToString()!, employee\_id: id, lastName: reader["lastName"].ToString()!, registeredAt: reader["registeredAt"].ToString()!);

user.id = reader["id"].ToString()!;

conn.Close();

return user;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//get user by id

public static User getById(string id)

{

try

{

string query = $"select \* from [user] where id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

User user = new User(User\_Type.Supplier, reader["firstName"].ToString()!, reader["username"].ToString()!, reader["mobile"].ToString()!, reader["email"].ToString()!, reader["address"].ToString()!, customer\_id: reader["customer\_id"].ToString(), lastName: reader["lastName"].ToString()!, registeredAt: reader["registeredAt"].ToString()!);

user.id = reader["id"].ToString()!;

conn.Close();

return user;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//updateByCustomerId

public static void updateByCustomerId(User user)

{

try

{

// user is a reserved keyword in SQL Server, so using [user] to avoid syntax issues

string query = $"update [user] set user\_type = @user\_type, firstName = @firstName, lastName = @lastName, username = @username, mobile = @mobile, email = @email, address = @address where customer\_id = @customer\_id;";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@user\_type", user.user\_type.ToString());

cmd.Parameters.AddWithValue("@firstName", user.firstName);

cmd.Parameters.AddWithValue("@lastName", user.lastName ?? (object)DBNull.Value);

cmd.Parameters.AddWithValue("@username", user.username);

cmd.Parameters.AddWithValue("@mobile", user.mobile);

cmd.Parameters.AddWithValue("@email", user.email);

cmd.Parameters.AddWithValue("@address", user.address);

cmd.Parameters.AddWithValue("@customer\_id", user.customer\_id);

conn.Open();

cmd.ExecuteNonQuery();

cmd.Dispose();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//updateByEmployeeId

public static void updateByEmployeeId(User user)

{

try

{

// user is a reserved keyword in SQL Server, so using [user] to avoid syntax issues

string query = $"update [user] set user\_type = @user\_type, firstName = @firstName, lastName = @lastName, username = @username, mobile = @mobile, email = @email, address = @address where employee\_id = @employee\_id;";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@user\_type", user.user\_type.ToString());

cmd.Parameters.AddWithValue("@firstName", user.firstName);

cmd.Parameters.AddWithValue("@lastName", user.lastName ?? (object)DBNull.Value);

cmd.Parameters.AddWithValue("@username", user.username);

cmd.Parameters.AddWithValue("@mobile", user.mobile);

cmd.Parameters.AddWithValue("@email", user.email);

cmd.Parameters.AddWithValue("@address", user.address);

cmd.Parameters.AddWithValue("@employee\_id", user.employee\_id);

conn.Open();

cmd.ExecuteNonQuery();

cmd.Dispose();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//updateBySupplierId

public static void updateBySupplierId(User user)

{

try

{

// user is a reserved keyword in SQL Server, so using [user] to avoid syntax issues

string query = $"update [user] set user\_type = @user\_type, firstName = @firstName, lastName = @lastName, username = @username, mobile = @mobile, email = @email, address = @address where supplier\_id = @supplier\_id;";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@user\_type", user.user\_type.ToString());

cmd.Parameters.AddWithValue("@firstName", user.firstName);

cmd.Parameters.AddWithValue("@lastName", user.lastName ?? (object)DBNull.Value);

cmd.Parameters.AddWithValue("@username", user.username);

cmd.Parameters.AddWithValue("@mobile", user.mobile);

cmd.Parameters.AddWithValue("@email", user.email);

cmd.Parameters.AddWithValue("@address", user.address);

cmd.Parameters.AddWithValue("@supplier\_id ", user.supplier\_id);

conn.Open();

cmd.ExecuteNonQuery();

cmd.Dispose();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//Order----------------->>>>

public class SqlOrder : SqlHelper

{

//adds a new Order and returns Order object with the id

//1) A user (customer (selling to)/supplier (buying from)) AND an employee (who is processing the order) AND type(in/out) are needed for Order

public static Order add(Order order)

{

try

{

//user is reserved word in sqlserver so we have to write it in [user] like this

string query = $"insert into [order] (user\_id, employee\_id, type, subTotal, tax, total) values (@user\_id, @employee\_id, @type, @subTotal, @tax, @total); select SCOPE\_IDENTITY();";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@user\_id", order.user\_id);

cmd.Parameters.AddWithValue("@employee\_id", order.employee\_id);

cmd.Parameters.AddWithValue("@type", order.type.ToString());

cmd.Parameters.AddWithValue("@subTotal", order.subTotal);

cmd.Parameters.AddWithValue("@tax", order.tax);

cmd.Parameters.AddWithValue("@total", order.total);

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

order.id = id;

cmd.Dispose();

conn.Close();

return order;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//getAll

public static List<Order> getAll()

{

try

{

List<Order> list = new List<Order>();

string query = $"select \* from [order];";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

Order order = new Order(reader["user\_id"].ToString()!, reader["employee\_id"].ToString()!, (DataModels.Type)Enum.Parse(typeof(DataModels.Type), reader["type"].ToString()!), Convert.ToDecimal(reader["subTotal"].ToString()), Convert.ToDecimal(reader["tax"].ToString()), Convert.ToDecimal(reader["total"].ToString()));

order.id = reader["id"].ToString()!;

list.Add(order);

}

cmd.Dispose();

conn.Close();

return list;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//Payment----------------->>>>

public class SqlPayment : SqlHelper

{

//adds a new Payment and returns Payment object with the id and createdAt

//1) A Order, User and ShippingAddress is needed for Payment object

public static Payment add(Payment payment)

{

try

{

string query = $"insert into payment (user\_id, order\_id, shippintAddress\_id, mode, status, type) values (@user\_id, @order\_id, @shippintAddress\_id, @mode, @status, @type); select SCOPE\_IDENTITY();";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@user\_id", payment.user\_id);

cmd.Parameters.AddWithValue("@order\_id", payment.order\_id);

cmd.Parameters.AddWithValue("@shippintAddress\_id", payment.shippingAddress\_id);

cmd.Parameters.AddWithValue("@mode", payment.mode.ToString());

cmd.Parameters.AddWithValue("@status", payment.status.ToString());

cmd.Parameters.AddWithValue("@type", payment.type.ToString());

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

payment.id = id;

//this extra work to refetch the record we added and pull the createdAt field

query = $"select createdAt from payment where id = {id}";

cmd = new SqlCommand(query, conn);

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

payment.createdAt = reader["createdAt"].ToString()!;

reader.Close();

conn.Close();

return payment;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//Update

public static Payment update(Payment payment)

{

try

{

string query = "update payment set user\_id = @user\_id, order\_id = @order\_id, shippingAddress\_id = @shippingAddress\_id, mode = @mode, status = @status, type = @type where id = @id;";

SqlCommand cmd = new SqlCommand(query, conn);

// Set up parameters for the SQL command

cmd.Parameters.AddWithValue("@user\_id", payment.user\_id);

cmd.Parameters.AddWithValue("@order\_id", payment.order\_id);

cmd.Parameters.AddWithValue("@shippingAddress\_id", payment.shippingAddress\_id);

cmd.Parameters.AddWithValue("@mode", payment.mode);

cmd.Parameters.AddWithValue("@status", payment.status);

cmd.Parameters.AddWithValue("@type", payment.type);

cmd.Parameters.AddWithValue("@id", payment.id);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

// Re-fetch the updated record's createdAt field, in case it was modified or for consistency

query = "select createdAt from payment where id = @id";

cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@id", payment.id);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

if (reader.Read())

{

payment.createdAt = reader["createdAt"].ToString()!;

}

reader.Close();

conn.Close();

return payment;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

public static Payment getByOrderId(string order\_id)

{

try

{

string query = $"select \* from payment where order\_id = {order\_id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

Payment payment = new Payment(reader["user\_id"].ToString()!, reader["order\_id"].ToString()!, reader["shippintAddress\_id"].ToString()!, (Mode)Enum.Parse(typeof(Mode), reader["mode"].ToString()!), (Status)Enum.Parse(typeof(Status), reader["status"].ToString()!), (DataModels.Type)Enum.Parse(typeof(DataModels.Type), reader["type"].ToString()!), createdAt: reader["createdAt"].ToString());

payment.id = reader["id"].ToString()!;

reader.Close();

conn.Close();

return payment;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//Product----------------->>>>

public class SqlProduct : SqlHelper

{

//adds a new Product and returns Product object with the id and createdAt

public static Product add(Product product)

{

try

{

string query = $"insert into product (title, description) values (@title, @description); select SCOPE\_IDENTITY();";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@title", product.title);

cmd.Parameters.AddWithValue("@description", product.description ?? (object)DBNull.Value);

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

product.id = id;

//this extra work to refetch the record we added and pull the createdAt field

query = $"select createdAt from product where id = {id}";

cmd = new SqlCommand(query, conn);

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

product.createdAt = reader["createdAt"].ToString()!;

reader.Close();

conn.Close();

return product;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//getAll

public static List<Product> getAll()

{

try

{

List<Product> list = new List<Product>();

string query = $"select \* from product;";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

Product product = new Product(reader["title"].ToString()!, reader["description"].ToString()!);

product.createdAt = reader["createdAt"].ToString();

product.id = reader["id"].ToString()!;

list.Add(product);

}

conn.Close();

return list;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//getById

public static Product getById(string id)

{

try

{

string query = $"select \* from product where id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

Product product = new Product(reader["title"].ToString()!, reader["description"].ToString()!);

product.createdAt = reader["createdAt"].ToString();

product.id = reader["id"].ToString()!;

conn.Close();

return product;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//delete

public static void delete(string id)

{

try

{

string query = $"delete from product where id = {id}";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//update

public static void update(Product product)

{

try

{

string query = $"update product set title = @title, description = @description where id = @id;";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@title", product.title);

cmd.Parameters.AddWithValue("@description", product.description ?? (object)DBNull.Value);

cmd.Parameters.AddWithValue("@id", product.id);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//Order\_Item----------------->>>>

public class SqlOrder\_Item : SqlHelper

{

//adds a new Order\_Item and returns Order\_Item object with the id

//product, Item and order are needed for this

public static Order\_Item add(Order\_Item order\_Item)

{

try

{

string query = $"insert into order\_item (product\_id, item\_id, order\_id, price, quantitiy, total\_price) values (@product\_id, @item\_id, @order\_id, @price, @quantitiy, @total\_price); select SCOPE\_IDENTITY();";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@product\_id", order\_Item.product\_id);

cmd.Parameters.AddWithValue("@item\_id", order\_Item.item\_id);

cmd.Parameters.AddWithValue("@order\_id", order\_Item.order\_id);

cmd.Parameters.AddWithValue("@price", order\_Item.price);

cmd.Parameters.AddWithValue("@quantitiy", order\_Item.quantity);

cmd.Parameters.AddWithValue("@total\_price", order\_Item.total\_price);

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

order\_Item.id = id;

conn.Close();

return order\_Item;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//Product\_Category----------------->>>>

public class SqlProduct\_Category : SqlHelper

{

//adds a new Product\_Category and returns Product\_Category object with the id

//category and product are needed for this

public static Product\_Category add(Product\_Category product\_Category)

{

try

{

string query = $"insert into product\_category (category\_id, product\_id) values (@category\_id, @product\_id); select SCOPE\_IDENTITY();";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@category\_id", product\_Category.category\_id);

cmd.Parameters.AddWithValue("@product\_id", product\_Category.product\_id);

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

product\_Category.id = id;

conn.Close();

return product\_Category;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//Category----------------->>>>

public class SqlCategory : SqlHelper

{

//adds a new Category and returns Category object with the id

public static Category add(Category category)

{

try

{

string query = $"insert into category (title, description) values (@title, @description); select SCOPE\_IDENTITY();";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@title", category.title);

cmd.Parameters.AddWithValue("@description", category.description);

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

category.id = id;

conn.Close();

return category;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

//Item----------------->>>>

public class SqlItem : SqlHelper

{

//adds a new Item and returns Item object with the id

public static Item add(Item item)

{

try

{

string query = $"insert into item (product\_id, brand\_id, supplier\_id, price, discount, quantity, stockValue, alarm\_quantity) values (@product\_id, @brand\_id, @supplier\_id, @price, @discount, @quantity, @stockValue, @alarm\_quantity); select SCOPE\_IDENTITY();";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@product\_id", item.product\_id);

cmd.Parameters.AddWithValue("@brand\_id", item.brand\_id);

cmd.Parameters.AddWithValue("@supplier\_id", item.supplier\_id);

cmd.Parameters.AddWithValue("@price", item.price);

cmd.Parameters.AddWithValue("@discount", item.discount);

cmd.Parameters.AddWithValue("@quantity", item.quantity);

cmd.Parameters.AddWithValue("@stockValue", item.stockValue);

cmd.Parameters.AddWithValue("@alarm\_quantity", item.alarm\_quantity);

conn.Open();

string id = cmd.ExecuteScalar().ToString()!;

item.id = id;

conn.Close();

return item;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//getAll

public static List<Item> getAll()

{

try

{

List<Item> list = new List<Item>();

string query = $"select \* from item;";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

Item item = new Item(reader["product\_id"].ToString()!, reader["brand\_id"].ToString()!, reader["supplier\_id"].ToString()!, Convert.ToDecimal(reader["price"].ToString()), Convert.ToInt32(reader["discount"].ToString()), Convert.ToInt32(reader["quantity"].ToString()), Convert.ToDecimal(reader["stockValue"].ToString()), Convert.ToInt64(reader["alarm\_quantity"].ToString()));

item.id = reader["id"].ToString()!;

list.Add(item);

}

conn.Close();

return list;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

public static Item getById(string id)

{

try

{

string query = $"select \* from item where id = {id};";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

Item item = new Item(reader["product\_id"].ToString()!, reader["brand\_id"].ToString()!, reader["supplier\_id"].ToString()!, Convert.ToDecimal(reader["price"].ToString()), Convert.ToInt32(reader["discount"].ToString()), Convert.ToInt32(reader["quantity"].ToString()), Convert.ToDecimal(reader["stockValue"].ToString()), Convert.ToInt64(reader["alarm\_quantity"].ToString()));

item.id = reader["id"].ToString()!;

conn.Close();

return item;

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//delete

public static void delete(string id)

{

try

{

string query = $"delete from item where id = {id}";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

//update

public static void update(Item item)

{

try

{

string query = @"update item

set product\_id = @product\_id,

brand\_id = @brand\_id,

supplier\_id = @supplier\_id,

price = @price,

discount = @discount,

quantity = @quantity,

stockValue = @stockValue,

alarm\_quantity = @alarm\_quantity

where id = @id";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@product\_id", item.product\_id);

cmd.Parameters.AddWithValue("@brand\_id", item.brand\_id);

cmd.Parameters.AddWithValue("@supplier\_id", item.supplier\_id);

cmd.Parameters.AddWithValue("@price", item.price);

cmd.Parameters.AddWithValue("@discount", item.discount);

cmd.Parameters.AddWithValue("@quantity", item.quantity);

cmd.Parameters.AddWithValue("@stockValue", item.stockValue);

cmd.Parameters.AddWithValue("@alarm\_quantity", item.alarm\_quantity);

cmd.Parameters.AddWithValue("@id", item.id); // Make sure item.id has the correct ID for the update

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

catch (Exception ex)

{

conn.Close();

throw new Exception(ex.Message);

}

}

}

}

**IMS MAIN**

**FORM1.CS**

using DataModels;

using DataAccessLayer;

namespace IMS\_Main

{

public partial class Dashboard : Form

{

//"admin" means admin else there will be employee id in this variable

public string loggedInAs;

public Dashboard(string loggedInAs)

{

InitializeComponent();

this.loggedInAs = loggedInAs;

//first the dashboard should appear

controlDashboard1.refreshProfit();

controlDashboard1.refreshBestCustomer();

controlDashboard1.refreshUserCount();

controlDashboard1.refreshBiggestOrderValue();

controlDashboard1.refreshInventoryOnAlarmQuantity();

controlDashboard1.refreshDate();

controlDashboard1.BringToFront();

button4.Text = "Orders";

}

private void button5\_Click(object sender, EventArgs e)

{

label2.Text = "Dashboard";

controlDashboard1.refreshProfit();

controlDashboard1.refreshBestCustomer();

controlDashboard1.refreshUserCount();

controlDashboard1.refreshBiggestOrderValue();

controlDashboard1.refreshInventoryOnAlarmQuantity();

controlDashboard1.refreshDate();

controlDashboard1.BringToFront();

}

private void button1\_Click(object sender, EventArgs e)

{

label2.Text = "Inventory";

controlinventory1.Controlinventory\_Load(sender, e);

controlinventory1.BringToFront();

}

private void button2\_Click(object sender, EventArgs e)

{

label2.Text = "Products";

controlProducts1.BringToFront();

}

private void button3\_Click(object sender, EventArgs e)

{

label2.Text = "Users";

controlUsers1.BringToFront();

}

private void button4\_Click(object sender, EventArgs e)

{

label2.Text = "Orders";

controlTransactions1.ControlTransactions\_Load(sender, e);

controlTransactions1.BringToFront();

}

private void Dashboard\_Load(object sender, EventArgs e)

{

//setting the name of logged in Employee or Admin

if(loggedInAs == "admin")

{

label3.Text = "Admin";

}

else

{

label3.Text = SqlUser.getByEmployeeId(loggedInAs).firstName;

}

//default self customer add

List<Customer> customers = SqlCustomer.getAll();

bool flag = false;

foreach (Customer customer in customers)

{

if (customer.email == "self@gmail.com")

{

flag = true;

break;

}

}

if (flag == false)

{

//else create one

SqlCustomer.makeself();

User user = SqlUser.add(new User(User\_Type.Customer, "self", "self", "9999999999", "self@gmail.com", "self", customer\_id: SqlCustomer.getSelfId()));

}

flag = false;

//its shipping address also

List<ShippingAddress> shippingAddresses = SqlShippingAddress.getAll();

foreach (ShippingAddress shippingAddresse in shippingAddresses)

{

if (shippingAddresse.customer\_id == SqlCustomer.getSelfId())

{

flag = true;

break;

}

}

if(flag == false)

{

SqlShippingAddress.makeSelf();

}

}

}

}

**LOGIN FORM**

using DataAccessLayer;

using DataModels;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace IMS\_Main

{

public partial class LoginForm : Form

{

public static string loggedInAs;

Captcha captcha;

public LoginForm()

{

InitializeComponent();

captcha = SqlCaptcha.fetchRandomCaptcha(SqlCaptcha.fetchNoOfCaptchas());

pictureBox1.Image = Image.FromFile(captcha.Path);

}

public void refreshForm()

{

captcha = SqlCaptcha.fetchRandomCaptcha(SqlCaptcha.fetchNoOfCaptchas());

pictureBox1.Image = Image.FromFile(captcha.Path);

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

textBox4.Clear();

textBox5.Clear();

}

//Admin Login

private void button1\_Click(object sender, EventArgs e)

{

if(textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "")

{

label8.Text = "Fill All Details First";

label8.Visible = true;

return;

}

string userCaptcha = textBox1.Text;

if (userCaptcha != captcha.Text)

{

label8.Text = "Wrong Captcha Text";

label8.Visible = true;

refreshForm();

return;

}

if (textBox2.Text != "admin" || textBox3.Text != "123")

{

label8.Text = "Wrong Credentials";

label8.Visible = true;

refreshForm();

return;

}

Dashboard mainForm = new Dashboard("admin");

loggedInAs = "admin";

mainForm.ShowDialog();

this.Close();

}

//Employee Login

private void button2\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "" || textBox4.Text == "" || textBox5.Text == "")

{

label8.Text = "Fill All Details First";

label8.Visible = true;

return;

}

string userCaptcha = textBox1.Text;

if (userCaptcha != captcha.Text)

{

label8.Text = "Wrong Captcha Text";

label8.Visible = true;

refreshForm();

return;

}

List<Employee> employees = SqlEmployee.getAll();

foreach (Employee employee in employees)

{

User user = SqlUser.getByEmployeeId(employee.id!);

if(textBox5.Text == user.username)

{

if(textBox4.Text == employee.passwordHash)

{

Dashboard mainForm = new Dashboard(employee.id!);

loggedInAs = employee.id!;

mainForm.ShowDialog();

this.Close();

}

}

}

label8.Text = "Wrong Credentials";

label8.Visible = true;

refreshForm();

}

}

}

**CONTROL DASHBOARD**

using DataAccessLayer;

using DataModels;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace IMS\_Main

{

public partial class ControlDashboard : UserControl

{

public ControlDashboard()

{

InitializeComponent();

refreshProfit();

refreshBestCustomer();

refreshUserCount();

refreshBiggestOrderValue();

refreshInventoryOnAlarmQuantity();

refreshDate();

}

public void refreshProfit()

{

decimal profit = 0;

List<Order> orders = SqlOrder.getAll();

foreach (Order order in orders)

{

if(order.type == DataModels.Type.Out)

{

profit += order.total;

}

}

label3.Text = $"Rs {profit}";

}

public void refreshBestCustomer()

{

label7.Text = "Not Known Yet";

List<Order> orders = SqlOrder.getAll();

if (orders.Count > 0)

{

decimal value = 0;

Order? biggestOrder = null;

foreach (Order order in orders)

{

if(order.type == DataModels.Type.Out && order.total > value)

{

biggestOrder = order;

value = order.total;

}

}

if (biggestOrder != null)

{

User user = SqlUser.getById(biggestOrder.user\_id);

label7.Text = $"{user.firstName} {user.lastName}";

}

}

}

public void refreshUserCount()

{

List<Customer> customers = SqlCustomer.getAll();

List<Supplier> suppliers = SqlSupplier.getAll();

List<Employee> employees = SqlEmployee.getAll();

label11.Text = customers.Count.ToString();

label12.Text = suppliers.Count.ToString();

label13.Text = employees.Count.ToString();

}

public void refreshBiggestOrderValue()

{

label1.Text = "Rs 0";

List<Order> orders = SqlOrder.getAll();

if (orders.Count > 0)

{

decimal value = 0;

Order? biggestOrder = null;

foreach (Order order in orders)

{

if (order.type == DataModels.Type.Out && order.total > value)

{

biggestOrder = order;

value = order.total;

}

}

if (biggestOrder != null)

{

label1.Text = $"Rs {biggestOrder.total}";

}

}

}

public void refreshInventoryOnAlarmQuantity()

{

List<Item> items = SqlItem.getAll();

List<Item> alarmList = new List<Item>();

foreach (Item it in items)

{

if(it.quantity <= it.alarm\_quantity)

{

alarmList.Add(it);

}

}

dataGridView1.DataSource = alarmList;

}

public void refreshDate()

{

DateTime today = DateTime.Today;

label5.Text = today.ToString("d");

}

}

}

**CONTROL INVENTORY**

using DataAccessLayer;

using DataModels;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace IMS\_Main

{

public partial class Controlinventory : UserControl

{

List<Item> items;

Item? item;

public Controlinventory()

{

InitializeComponent();

}

//initial fetch to load data

public void Controlinventory\_Load(object sender, EventArgs e)

{

comboBox1.Items.Clear();

//comboBox2.Items.Clear();

comboBox3.Items.Clear();

items = SqlItem.getAll();

this.BackColor = Color.FromArgb(12, 0, 50); //had to do this coz it wasn't changing from ui settings

dataGridView1.DataSource = items;

//List<Brand> brands = SqlBrand.getAll();

List<Product> products = SqlProduct.getAll();

List<Supplier> suppliers = SqlSupplier.getAll();

foreach (var item in products)

{

comboBox1.Items.Add(item);

}

//foreach (var item in brands)

//{

// comboBox2.Items.Add(item.id!);

//}

foreach (var item in suppliers)

{

comboBox3.Items.Add(item);

}

}

//when a row is selected

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

// Check if the row index is valid (e.g., avoid header row clicks)

if (e.RowIndex >= 0)

{

// Get the clicked row

DataGridViewRow selectedRow = dataGridView1.Rows[e.RowIndex];

textBox1.Text = SqlProduct.getById(selectedRow.Cells[0].Value.ToString()!).ToString(); // Value of the first cell

comboBox1.Text = selectedRow.Cells[1].Value.ToString();

//comboBox2.Text = selectedRow.Cells[2].Value.ToString();

comboBox3.Text = SqlSupplier.getById(selectedRow.Cells[3].Value.ToString()!).ToString();

textBox8.Text = selectedRow.Cells[4].Value.ToString();

textBox7.Text = selectedRow.Cells[5].Value.ToString();

textBox6.Text = selectedRow.Cells[6].Value.ToString();

textBox5.Text = selectedRow.Cells[7].Value.ToString();

textBox9.Text = selectedRow.Cells[8].Value.ToString();

}

}

//Delete

private void button3\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "" || comboBox1.Text == "" || comboBox3.Text == "") return;

try

{

SqlItem.delete(textBox1.Text);

MessageBox.Show("Item deleted");

//refresh grid

items = SqlItem.getAll();

dataGridView1.DataSource = items;

}

catch(Exception ex)

{

label10.Text = ex.Message;

}

}

//Add

private void button1\_Click(object sender, EventArgs e)

{

try

{

if (textBox8.Text == "" || comboBox1.Text == "" || comboBox3.Text == "") return;

Item item = new Item(comboBox1.Text[0].ToString(), "1", comboBox3.Text[0].ToString(), Convert.ToDecimal(textBox8.Text), Convert.ToInt32(textBox7.Text), Convert.ToInt32(textBox6.Text), Convert.ToDecimal(textBox5.Text), Convert.ToInt64(textBox9.Text));

SqlItem.add(item);

Controlinventory\_Load(sender, e); //reloads OR refreshes not working

items = SqlItem.getAll();

dataGridView1.DataSource = items;

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

//update

private void button2\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "" || comboBox1.Text == "" || comboBox3.Text == "") return;

Item item = new Item(comboBox1.Text[0].ToString(), "1", comboBox3.Text[0].ToString(), Convert.ToDecimal(textBox8.Text), Convert.ToInt32(textBox7.Text), Convert.ToInt32(textBox6.Text), Convert.ToDecimal(textBox5.Text), Convert.ToInt64(textBox9.Text));

item.id = textBox1.Text;

try

{

SqlItem.update(item);

//refreshing

items = SqlItem.getAll();

dataGridView1.DataSource = items;

MessageBox.Show("Item Updated Successfully");

}

catch (Exception ex)

{

label10.Text = ex.Message;

}

}

}

}

**CONTROL PRODUCTS**

using DataAccessLayer;

using DataModels;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace IMS\_Main

{

public partial class ControlProducts : UserControl

{

List<Product> products;

public ControlProducts()

{

InitializeComponent();

products = SqlProduct.getAll();

}

private void ControlProducts\_Load(object sender, EventArgs e)

{

dataGridView1.DataSource = products;

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

// Check if the row index is valid (e.g., avoid header row clicks)

if (e.RowIndex >= 0)

{

// Get the clicked row

DataGridViewRow selectedRow = dataGridView1.Rows[e.RowIndex];

textBox1.Text = selectedRow.Cells[0].Value.ToString(); // Value of the first cell

textBox8.Text = selectedRow.Cells[1].Value.ToString();

richTextBox1.Text = selectedRow.Cells[2].Value.ToString();

richTextBox2.Text = selectedRow.Cells[3].Value.ToString();

}

}

//add

private void button1\_Click(object sender, EventArgs e)

{

if (textBox8.Text == "") return;

Product product = new Product(textBox8.Text, richTextBox1.Text);

try

{

product = SqlProduct.add(product);

products = SqlProduct.getAll();

dataGridView1.DataSource = products;

MessageBox.Show($"Product added with id: {product.id}");

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

//Delete

private void button3\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "") return;

try

{

SqlProduct.delete(textBox1.Text);

MessageBox.Show("Item deleted");

//refresh grid

products = SqlProduct.getAll();

dataGridView1.DataSource = products;

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

//update

private void button2\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "") return;

Product product = new Product(textBox8.Text, richTextBox1.Text);

product.id = textBox1.Text;

try

{

SqlProduct.update(product);

//refreshing

products = SqlProduct.getAll();

dataGridView1.DataSource = products;

MessageBox.Show("Item Updated Successfully");

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

}

}

**CONTROL ORDERS**

using DataAccessLayer;

using DataModels;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace IMS\_Main

{

public partial class ControlTransactions : UserControl

{

private List<Order> orders;

private Item? item = null; //this is the item selected from the list

List<Item> items = SqlItem.getAll(); //initial list to show available items to users

//binding list automatically updated UIs like listbox if its datasource is set to this bindinglist

private BindingList<UserItem> userItems = new BindingList<UserItem>(); //this is displayed on listBox whatever user has added

public ControlTransactions()

{

InitializeComponent();

}

public void ControlTransactions\_Load(object sender, EventArgs e)

{

//fill comboboxes with values

comboBox3.DataSource = Enum.GetValues(typeof(DataModels.Type));

List<Item> items = SqlItem.getAll();

comboBox1.Items.Clear();

foreach (var item in items)

{

comboBox1.Items.Add(item.id!);

}

listBox1.DataSource = userItems;

comboBox5.DataSource = SqlEmployee.getAll();

//dataGridView filling

refreshOrderData();

}

private void refreshOrderData()

{

// Set up columns for selective properties if not already done

dataGridView1.Columns.Clear(); // Clear existing columns if needed

// Add columns for selective properties

dataGridView1.Columns.Add("order\_id", "order\_id");

dataGridView1.Columns.Add("customer\_id", "customer\_id");

dataGridView1.Columns.Add("type", "type");

dataGridView1.Columns.Add("total", "total");

dataGridView1.Columns.Add("shippingAddress", "shippingAddress");

dataGridView1.Columns.Add("createdAt", "createdAt");

orders = SqlOrder.getAll();

foreach (Order order in orders)

{

User user = SqlUser.getById(order.user\_id!);

Payment payment = SqlPayment.getByOrderId(order.id!);

ShippingAddress shippingAddress = SqlShippingAddress.getById(payment.shippingAddress\_id);

dataGridView1.Rows.Add(order.id, user.customer\_id!, payment.type.ToString(), order.total, $"{shippingAddress.city}, {shippingAddress.state}, {shippingAddress.country}", payment.createdAt);

}

}

//type In/Out

private void comboBox3\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox3.Text == "") return;

if (comboBox3.Text == "In")

{

comboBox2.Text = "self";

comboBox4.Text = "self";

comboBox2.Enabled = false;

comboBox4.Enabled = false;

}

else if (comboBox3.Text == "Out")

{

comboBox2.Enabled = true;

comboBox4.Enabled = true;

comboBox2.DataSource = SqlCustomer.getAll();

}

}

//when customer is selected so show all his shipping addresses

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox2.Text == "") return;

comboBox4.DataSource = SqlShippingAddress.getMany(comboBox2.Text[0].ToString());

}

//when item is selected then fetch its price

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.Text == "") return;

item = getItemFromMyItemList(comboBox1.Text);

textBox2.Text = item.price.ToString();

}

//when add item button is clicked

private void button3\_Click(object sender, EventArgs e)

{

try

{

if (item == null) return;

int quantity = Convert.ToInt32(maskedTextBox1.Text);

if (quantity > item!.quantity) throw new Exception("Not enough quantity in Inventory");

UserItem userItem = new UserItem(item.product\_id, item.id!, item.price.ToString(), maskedTextBox1.Text, textBox3.Text);

userItems.Add(userItem);

item.quantity -= quantity;

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

//when remove item is clicked

private void button2\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedItem == null) return;

UserItem userItem = (UserItem)listBox1.SelectedItem;

userItems.Remove(userItem);

item = getItemFromMyItemList(userItem.item\_id);

item.quantity += Convert.ToInt32(userItem.quantity);

}

//when quantity is changed

private void maskedTextBox1\_TextChanged(object sender, EventArgs e)

{

try

{

if (maskedTextBox1.Text == "")

{

textBox3.Text = "";

return;

}

decimal quantity = Convert.ToDecimal(maskedTextBox1.Text);

if (quantity <= 0) throw new Exception("Quanitiy should be at least 1");

decimal price = Convert.ToDecimal(textBox2.Text);

textBox3.Text = (quantity \* price).ToString();

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

//extra utility function

public Item getItemFromMyItemList(string id)

{

foreach (Item myItem in items)

{

if (myItem.id == id) return myItem;

}

return items[0]; //you will never know why I wrote this line XD (it will never even execute lol)

}

//Add Order

private void button1\_Click(object sender, EventArgs e)

{

try

{

if (listBox1.Items.Count == 0 || comboBox2.Text == "" || comboBox3.Text == "" || comboBox4.Text == "" || comboBox5.Text == "") return;

User user;

Payment payment;

Order order;

if (comboBox3.Text == "In")

{

user = SqlUser.getByCustomerId(SqlCustomer.getSelfId());

order = new Order(user.id!, comboBox5.Text[0].ToString(), (DataModels.Type)Enum.Parse(typeof(DataModels.Type), comboBox3.Text), Convert.ToDecimal(textBox3.Text), 0, Convert.ToDecimal(textBox3.Text));

order = SqlOrder.add(order);

payment = new Payment(user.id!, order.id!, SqlShippingAddress.getSelfId(), Mode.Online, Status.Finished, (DataModels.Type)Enum.Parse(typeof(DataModels.Type), comboBox3.Text));

}

else

{

user = SqlUser.getByCustomerId(comboBox2.Text[0].ToString());

order = new Order(user.id!, comboBox5.Text[0].ToString(), (DataModels.Type)Enum.Parse(typeof(DataModels.Type), comboBox3.Text), Convert.ToDecimal(textBox3.Text), 0, Convert.ToDecimal(textBox3.Text));

order = SqlOrder.add(order);

payment = new Payment(user.id!, order.id!, comboBox4.Text[0].ToString(), Mode.Online, Status.Finished, (DataModels.Type)Enum.Parse(typeof(DataModels.Type), comboBox3.Text));

}

payment = SqlPayment.add(payment);

foreach (UserItem item in userItems)

{

Order\_Item orderItem = new Order\_Item(item.product\_id, item.item\_id, order.id!, Convert.ToDecimal(item.price), Convert.ToInt64(item.quantity), Convert.ToDecimal(item.totalPrice));

SqlOrder\_Item.add(orderItem);

Item inventoryQuantityUpdate = SqlItem.getById(orderItem.item\_id);

//if sold to customer then reduce quantity from inventory

if (comboBox3.Text == "Out")

{

inventoryQuantityUpdate.quantity -= orderItem.quantity;

SqlItem.update(inventoryQuantityUpdate);

}

//if bought from supplier then increase quantity in inventory

else

{

inventoryQuantityUpdate.quantity += orderItem.quantity;

SqlItem.update(inventoryQuantityUpdate);

}

}

refreshOrderData();

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

}

//temporary class for showing items added by user in the list

public class UserItem {

public string product\_id;

public string item\_id;

public string price;

public string quantity;

public string totalPrice;

public UserItem(string product\_id, string item\_id, string price, string quantity, string totalPrice)

{

this.product\_id = product\_id;

this.item\_id = item\_id;

this.price = price;

this.quantity = quantity;

this.totalPrice = totalPrice;

}

override

public string ToString()

{

Product product = SqlProduct.getById(product\_id);

return $"{item\_id}) {product.title}: {totalPrice}";

}

}

}

**CONTROL USERS**

using DataAccessLayer;

using DataModels;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace IMS\_Main

{

public partial class ControlUsers : UserControl

{

List<Employee> employees;

List<Supplier> suppliers;

List<Customer> customers;

string currentUserType = "";

public ControlUsers()

{

InitializeComponent();

employees = SqlEmployee.getAll();

suppliers = SqlSupplier.getAll();

customers = SqlCustomer.getAll();

}

private void clearFields()

{

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

textBox4.Clear();

textBox6.Clear();

maskedTextBox1.Clear();

textBox8.Clear();

textBox9.Clear();

textBox10.Clear();

}

private void refreshEmployeeData()

{

// Set up columns for selective properties if not already done

dataGridView1.Columns.Clear(); // Clear existing columns if needed

// Add columns for selective properties

dataGridView1.Columns.Add("employee\_id", "employee\_id");

dataGridView1.Columns.Add("firstName", "firstNameName");

dataGridView1.Columns.Add("lastName", "lastName");

dataGridView1.Columns.Add("username", "username");

dataGridView1.Columns.Add("mobile", "mobile");

dataGridView1.Columns.Add("email", "email");

dataGridView1.Columns.Add("address", "address");

dataGridView1.Columns.Add("registeredAt", "registeredAt");

dataGridView1.Columns.Add("salary", "salary");

dataGridView1.Columns.Add("role", "role");

employees = SqlEmployee.getAll();

foreach (Employee employee in employees)

{

User user = SqlUser.getByEmployeeId(employee.id!);

dataGridView1.Rows.Add(employee.id, user.firstName, user.lastName, user.username, user.mobile, user.email, user.address, user.registeredAt, employee.salary, employee.role.ToString());

}

}

//Employee button

private void button3\_Click(object sender, EventArgs e)

{

try

{

if (LoginForm.loggedInAs != "admin")

{

throw new Exception("Only for Admin");

}

clearFields();

//showing extra fields of employee

label10.Visible = true;

label12.Visible = true;

textBox10.Visible = true;

textBox4.Visible = true;

button7.Visible = false; //shipping address for only customer

//filling type combobox with Customer types

comboBox3.DataSource = Enum.GetValues(typeof(Role));

currentUserType = "employee";

refreshEmployeeData();

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

private void refreshSupplierData()

{

// Set up columns for selective properties if not already done

dataGridView1.Columns.Clear(); // Clear existing columns if needed

dataGridView1.Columns.Add("supplier\_id", "supplier\_id");

dataGridView1.Columns.Add("firstName", "firstNameName");

dataGridView1.Columns.Add("lastName", "lastName");

dataGridView1.Columns.Add("username", "username");

dataGridView1.Columns.Add("mobile", "mobile");

dataGridView1.Columns.Add("email", "email");

dataGridView1.Columns.Add("address", "address");

dataGridView1.Columns.Add("registeredAt", "registeredAt");

dataGridView1.Columns.Add("supplier\_type", "supplier\_type");

suppliers = SqlSupplier.getAll();

foreach (Supplier supplier in suppliers)

{

User user = SqlUser.getBySupplierId(supplier.id!);

dataGridView1.Rows.Add(supplier.id, user.firstName, user.lastName, user.username, user.mobile, user.email, user.address, user.registeredAt, supplier.supplier\_type.ToString());

}

}

//Supplier button

private void button2\_Click(object sender, EventArgs e)

{

clearFields();

//hiding extra fields of employee

label10.Visible = false;

label12.Visible = false;

textBox10.Visible = false;

textBox4.Visible = false;

button7.Visible = false; //shipping address for only customer

//filling type combobox with Customer types

comboBox3.DataSource = Enum.GetValues(typeof(Supplier\_Type));

currentUserType = "supplier";

refreshSupplierData();

}

private void refreshCustomerData()

{

// Set up columns for selective properties if not already done

dataGridView1.Columns.Clear(); // Clear existing columns if needed

dataGridView1.Columns.Add("customer\_id", "customer\_id");

dataGridView1.Columns.Add("firstName", "firstNameName");

dataGridView1.Columns.Add("lastName", "lastName");

dataGridView1.Columns.Add("username", "username");

dataGridView1.Columns.Add("mobile", "mobile");

dataGridView1.Columns.Add("email", "email");

dataGridView1.Columns.Add("address", "address");

dataGridView1.Columns.Add("registeredAt", "registeredAt");

dataGridView1.Columns.Add("customer\_type", "customer\_type");

customers = SqlCustomer.getAll();

foreach (Customer customer in customers)

{

User user = SqlUser.getByCustomerId(customer.id!);

dataGridView1.Rows.Add(customer.id, user.firstName, user.lastName, user.username, user.mobile, user.email, user.address, user.registeredAt, customer.customer\_type.ToString());

}

}

//Customer button

private void button1\_Click(object sender, EventArgs e)

{

clearFields();

//hiding extra fields of employee

label10.Visible = false;

label12.Visible = false;

textBox10.Visible = false;

textBox4.Visible = false;

button7.Visible = true; //shipping address for only customer

//filling type combobox with Customer types

comboBox3.DataSource = Enum.GetValues(typeof(Customer\_Type));

currentUserType = "customer";

refreshCustomerData();

}

//row is clicked

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

//if user is customer

if (currentUserType == "customer")

{

// Check if the row index is valid (e.g., avoid header row clicks)

if (e.RowIndex >= 0)

{

// Get the clicked row

DataGridViewRow selectedRow = dataGridView1.Rows[e.RowIndex];

string id = selectedRow.Cells[0].Value.ToString()!; // Value of the first cell

Customer customer = SqlCustomer.getById(id);

User user = SqlUser.getByCustomerId(id);

textBox1.Text = customer.id;

textBox8.Text = user.firstName;

textBox2.Text = user.lastName;

textBox3.Text = user.username;

maskedTextBox1.Text = user.mobile;

textBox6.Text = user.email;

richTextBox1.Text = user.address;

textBox9.Text = user.registeredAt;

comboBox3.Text = customer.customer\_type.ToString();

}

}

//if user is supplier

else if (currentUserType == "supplier")

{

// Check if the row index is valid (e.g., avoid header row clicks)

if (e.RowIndex >= 0)

{

// Get the clicked row

DataGridViewRow selectedRow = dataGridView1.Rows[e.RowIndex];

string id = selectedRow.Cells[0].Value.ToString()!; // Value of the first cell

Supplier supplier = SqlSupplier.getById(id);

User user = SqlUser.getBySupplierId(id);

textBox1.Text = supplier.id;

textBox8.Text = user.firstName;

textBox2.Text = user.lastName;

textBox3.Text = user.username;

maskedTextBox1.Text = user.mobile;

textBox6.Text = user.email;

richTextBox1.Text = user.address;

textBox9.Text = user.registeredAt;

comboBox3.Text = supplier.supplier\_type.ToString();

}

}

//if user is employee

else if (currentUserType == "employee")

{

// Check if the row index is valid (e.g., avoid header row clicks)

if (e.RowIndex >= 0)

{

// Get the clicked row

DataGridViewRow selectedRow = dataGridView1.Rows[e.RowIndex];

string id = selectedRow.Cells[0].Value.ToString()!; // Value of the first cell

Employee employee = SqlEmployee.getById(id);

User user = SqlUser.getByEmployeeId(id);

textBox1.Text = employee.id;

textBox8.Text = user.firstName;

textBox2.Text = user.lastName;

textBox3.Text = user.username;

maskedTextBox1.Text = user.mobile;

textBox6.Text = user.email;

richTextBox1.Text = user.address;

textBox9.Text = user.registeredAt;

comboBox3.Text = employee.role.ToString();

textBox10.Text = employee.salary.ToString();

textBox4.Text = employee.passwordHash.ToString();

}

}

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

//Add button is clicked

private void button4\_Click(object sender, EventArgs e)

{

try

{

if (currentUserType == "customer")

{

if (textBox3.Text == "" || textBox6.Text == "" || maskedTextBox1.Text == "" || textBox8.Text == "" || richTextBox1.Text == "" || comboBox3.Text == "") throw new Exception("Fill all details");

Customer customer = new Customer(textBox6.Text, (Customer\_Type)Enum.Parse(typeof(Customer\_Type), comboBox3.Text));

customer = SqlCustomer.add(customer);

User user = new User(User\_Type.Customer, textBox8.Text, textBox3.Text, maskedTextBox1.Text, textBox6.Text, richTextBox1.Text, customer\_id: customer.id, lastName: textBox2.Text);

user = SqlUser.add(user);

MessageBox.Show($"Customer Added with ID: {customer.id}");

refreshCustomerData();

}

else if (currentUserType == "supplier")

{

if (textBox3.Text == "" || textBox6.Text == "" || maskedTextBox1.Text == "" || textBox8.Text == "" || richTextBox1.Text == "" || comboBox3.Text == "") throw new Exception("Fill all details");

Supplier supplier = new Supplier(textBox6.Text, (Supplier\_Type)Enum.Parse(typeof(Supplier\_Type), comboBox3.Text));

supplier = SqlSupplier.add(supplier);

User user = new User(User\_Type.Supplier, textBox8.Text, textBox3.Text, maskedTextBox1.Text, textBox6.Text, richTextBox1.Text, supplier\_id: supplier.id, lastName: textBox2.Text);

user = SqlUser.add(user);

MessageBox.Show($"Supplier Added with ID: {supplier.id}");

refreshSupplierData();

}

else if (currentUserType == "employee")

{

if (textBox3.Text == "" || textBox6.Text == "" || maskedTextBox1.Text == "" || textBox8.Text == "" || richTextBox1.Text == "" || comboBox3.Text == "" || textBox4.Text == "" || textBox10.Text == "") throw new Exception("Fill all details");

Employee employee = new Employee(Convert.ToDecimal(textBox10.Text), (Role)Enum.Parse(typeof(Role), comboBox3.Text), textBox4.Text);

employee = SqlEmployee.add(employee);

User user = new User(User\_Type.Supplier, textBox8.Text, textBox3.Text, maskedTextBox1.Text, textBox6.Text, richTextBox1.Text, employee\_id: employee.id, lastName: textBox2.Text);

user = SqlUser.add(user);

MessageBox.Show($"Employee Added with ID: {employee.id}");

refreshEmployeeData();

}

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

//delete button is clicked

private void button5\_Click(object sender, EventArgs e)

{

try

{

if (currentUserType == "customer")

{

if (textBox1.Text == "") return;

SqlCustomer.delete(textBox1.Text);

refreshCustomerData();

}

else if (currentUserType == "supplier")

{

if (textBox1.Text == "") return;

SqlSupplier.delete(textBox1.Text);

refreshSupplierData();

}

else if (currentUserType == "employee")

{

if (textBox1.Text == "" || textBox4.Text == "" || textBox10.Text == "") throw new Exception("Fill all details");

SqlEmployee.delete(textBox1.Text);

refreshEmployeeData();

}

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

//update button is clicked

private void button6\_Click(object sender, EventArgs e)

{

try

{

if (currentUserType == "customer")

{

if (textBox1.Text == "") return;

Customer customer = new Customer(textBox6.Text, (Customer\_Type)Enum.Parse(typeof(Customer\_Type), comboBox3.Text));

customer.id = textBox1.Text;

SqlCustomer.update(customer);

User user = new User(User\_Type.Customer, textBox8.Text, textBox3.Text, maskedTextBox1.Text, textBox6.Text, richTextBox1.Text, customer\_id: textBox1.Text, lastName: textBox2.Text);

SqlUser.updateByCustomerId(user);

MessageBox.Show("Customer Updated");

refreshCustomerData();

}

else if (currentUserType == "supplier")

{

if (textBox1.Text == "") return;

Supplier Supplier = new Supplier(textBox6.Text, (Supplier\_Type)Enum.Parse(typeof(Supplier\_Type), comboBox3.Text));

Supplier.id = textBox1.Text;

SqlSupplier.update(Supplier);

User user = new User(User\_Type.Supplier, textBox8.Text, textBox3.Text, maskedTextBox1.Text, textBox6.Text, richTextBox1.Text, supplier\_id: textBox1.Text, lastName: textBox2.Text);

SqlUser.updateBySupplierId(user);

MessageBox.Show("Supplier Updated");

refreshSupplierData();

}

else if (currentUserType == "employee")

{

Employee Employee = new Employee(Convert.ToDecimal(textBox10.Text), (Role)Enum.Parse(typeof(Role), comboBox3.Text), textBox4.Text);

Employee.id = textBox1.Text;

SqlEmployee.update(Employee);

User user = new User(User\_Type.Employee, textBox8.Text, textBox3.Text, maskedTextBox1.Text, textBox6.Text, richTextBox1.Text, employee\_id: textBox1.Text, lastName: textBox2.Text);

SqlUser.updateByEmployeeId(user);

MessageBox.Show("Employee Updated");

refreshEmployeeData();

}

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

private void button7\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "") return;

ShippingAddressesForm shippingAddressesForm = new ShippingAddressesForm(textBox1.Text);

shippingAddressesForm.ShowDialog();

}

}

}

**LOADING FORM**

using DataAccessLayer;

using DataModels;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace IMS\_Main

{

public partial class ShippingAddressesForm : Form

{

string customer\_id;

public ShippingAddressesForm(string customer\_id)

{

InitializeComponent();

this.customer\_id = customer\_id;

}

private void ShippingAddressesForm\_Load(object sender, EventArgs e)

{

dataGridView1.DataSource = SqlShippingAddress.getMany(customer\_id);

User user = SqlUser.getByCustomerId(customer\_id);

label6.Text = $"{user.firstName} {user.lastName}";

}

//Row is clicked

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

// Check if the row index is valid (e.g., avoid header row clicks)

if (e.RowIndex >= 0)

{

// Get the clicked row

DataGridViewRow selectedRow = dataGridView1.Rows[e.RowIndex];

textBox4.Text = selectedRow.Cells[0].Value.ToString()!; //id

textBox5.Text = selectedRow.Cells[1].Value.ToString()!; //customer\_id

textBox1.Text = selectedRow.Cells[2].Value.ToString()!; //city

textBox2.Text = selectedRow.Cells[3].Value.ToString()!; //state

textBox3.Text = selectedRow.Cells[4].Value.ToString()!; //country

richTextBox1.Text = selectedRow.Cells[5].Value.ToString()!; //more

}

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

//Add button is clicked

private void button4\_Click(object sender, EventArgs e)

{

try

{

if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "") return;

ShippingAddress shippingAddress = new ShippingAddress(customer\_id, textBox1.Text, textBox2.Text, textBox3.Text, more: richTextBox1.Text);

//add and refresh UI

SqlShippingAddress.add(shippingAddress);

dataGridView1.DataSource = SqlShippingAddress.getMany(customer\_id);

MessageBox.Show("Address Added Successfully");

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

//delete button is clicked

private void button5\_Click(object sender, EventArgs e)

{

try

{

if (textBox4.Text == "") return;

//add and refresh UI

SqlShippingAddress.delete(textBox4.Text);

dataGridView1.DataSource = SqlShippingAddress.getMany(customer\_id);

MessageBox.Show("Address Deleted Successfully");

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

//update button is clicked

private void button6\_Click(object sender, EventArgs e)

{

try

{

if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "" || textBox4.Text == "" || textBox5.Text == "") return;

ShippingAddress shippingAddress = new ShippingAddress(textBox5.Text, textBox1.Text, textBox2.Text, textBox3.Text, more: richTextBox1.Text);

shippingAddress.id = textBox4.Text;

//update and refresh UI

SqlShippingAddress.update(shippingAddress);

dataGridView1.DataSource = SqlShippingAddress.getMany(customer\_id);

MessageBox.Show("Address Updated Successfully");

}

catch (Exception ex)

{

label11.Text = ex.Message;

}

}

private void label9\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**SQL QUERIES**

create database Inventory\_Management\_System;

use Inventory\_Management\_System;

create table captcha (

id int primary key identity(1, 1),

text nvarchar(10),

path nvarchar(100)

);

insert into captcha(text, path)

values

('TXGAP', 'images\image1.png'),

('MLPSY', 'images\image2.png'),

('NQCLA', 'images\image3.png')

CREATE TABLE "item"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"product\_id" BIGINT NOT NULL,

"brand\_id" BIGINT NOT NULL,

"supplier\_id" BIGINT NULL,

"price" BIGINT NOT NULL,

"discount" INT NOT NULL,

"quantity" BIGINT NOT NULL,

"stockValue" BIGINT NOT NULL,

"alarm\_quantity" BIGINT NOT NULL

);

ALTER TABLE

"item" ADD CONSTRAINT "item\_id\_primary" PRIMARY KEY("id");

CREATE TABLE "payment"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"user\_id" BIGINT NULL,

"order\_id" BIGINT NOT NULL,

"shippintAddress\_id" BIGINT NULL,

"mode" NVARCHAR(255) CHECK

("mode" IN(N'Online', N'Cod')) NOT NULL,

"status" NVARCHAR(255)

CHECK

(

"status" IN(N'Pending', N'Finished', N'Failed')

) NOT NULL,

"createdAt" DATETIME NOT NULL DEFAULT GETDATE(),

"type" NVARCHAR(255)

CHECK

("type" IN(N'In', N'Out')) NOT NULL

);

ALTER TABLE

"payment" ADD CONSTRAINT "payment\_id\_primary" PRIMARY KEY("id");

CREATE TABLE "order"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"user\_id" BIGINT NULL,

"employee\_id" BIGINT NULL,

"type" NVARCHAR(255) CHECK

("type" IN(N'In', N'Out')) NOT NULL,

"subTotal" BIGINT NOT NULL,

"tax" BIGINT NOT NULL,

"total" BIGINT NOT NULL

);

ALTER TABLE

"order" ADD CONSTRAINT "order\_id\_primary" PRIMARY KEY("id");

CREATE TABLE "shippingAddress"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"customer\_id" BIGINT NOT NULL,

"city" NVARCHAR(255) NOT NULL,

"state" NVARCHAR(255) NOT NULL,

"country" NVARCHAR(255) NOT NULL,

"more" NVARCHAR(255) NULL

);

ALTER TABLE

"shippingAddress" ADD CONSTRAINT "shippingaddress\_id\_primary" PRIMARY KEY("id");

CREATE TABLE "product"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"title" NVARCHAR(255) NOT NULL,

"description" NVARCHAR(255) NULL,

"createdAt" DATETIME NOT NULL DEFAULT GETDATE()

);

ALTER TABLE

"product" ADD CONSTRAINT "product\_id\_primary" PRIMARY KEY("id");

CREATE TABLE "customer"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"email" NVARCHAR(255) NOT NULL,

"customer\_type" NVARCHAR(255) CHECK

(

"customer\_type" IN(N'Rich', N'Poor', N'Medium')

) NOT NULL

);

ALTER TABLE

"customer" ADD CONSTRAINT "customer\_id\_primary" PRIMARY KEY("id");

CREATE UNIQUE INDEX "customer\_email\_unique" ON

"customer"("email");

CREATE TABLE "brand"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"title" NVARCHAR(255) NOT NULL,

"summary" NVARCHAR(255) NULL,

"popularity" NVARCHAR(255) CHECK

(

"popularity" IN(N'Low', N'Medium', N'High')

) NOT NULL

);

insert into brand values('Samsung', 'Korean company', 'High');

ALTER TABLE

"brand" ADD CONSTRAINT "brand\_id\_primary" PRIMARY KEY("id");

CREATE TABLE "employee"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"salary" BIGINT NOT NULL,

"role" NVARCHAR(255) CHECK

("role" IN(N'Manager', N'Sales')) NOT NULL,

"passwordHash" NVARCHAR(255) NOT NULL

);

ALTER TABLE

"employee" ADD CONSTRAINT "employee\_id\_primary" PRIMARY KEY("id");

CREATE TABLE "order\_item"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"product\_id" BIGINT NULL,

"item\_id" BIGINT NULL,

"order\_id" BIGINT NOT NULL,

"price" BIGINT NOT NULL,

"quantitiy" BIGINT NOT NULL,

"total\_price" BIGINT NOT NULL

);

ALTER TABLE

"order\_item" ADD CONSTRAINT "order\_item\_id\_primary" PRIMARY KEY("id");

CREATE TABLE "product\_category"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"category\_id" BIGINT NOT NULL,

"product\_id" BIGINT NOT NULL

);

ALTER TABLE

"product\_category" ADD CONSTRAINT "product\_category\_id\_primary" PRIMARY KEY("id");

CREATE TABLE "supplier"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"email" NVARCHAR(255) NOT NULL,

"supplier\_type" NVARCHAR(255) CHECK

(

"supplier\_type" IN(N'Trusted', N'New')

) NOT NULL

);

ALTER TABLE

"supplier" ADD CONSTRAINT "supplier\_id\_primary" PRIMARY KEY("id");

CREATE UNIQUE INDEX "supplier\_email\_unique" ON

"supplier"("email");

CREATE TABLE "category"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"title" NVARCHAR(255) NOT NULL,

"description" NVARCHAR(255) NULL

);

ALTER TABLE

"category" ADD CONSTRAINT "category\_id\_primary" PRIMARY KEY("id");

CREATE TABLE "loginSession"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"employee\_id" BIGINT NOT NULL,

"loggedInAt" DATETIME NOT NULL DEFAULT GETDATE()

);

ALTER TABLE

"loginSession" ADD CONSTRAINT "loginsession\_id\_primary" PRIMARY KEY("id");

CREATE TABLE "user"(

"id" BIGINT NOT NULL IDENTITY(1, 1),

"supplier\_id" BIGINT NULL,

"customer\_id" BIGINT NULL,

"employee\_id" BIGINT NULL,

"user\_type" NVARCHAR(255) CHECK

(

"user\_type" IN(

N'Customer',

N'Supplier',

N'Employee'

)

) NOT NULL,

"firstName" NVARCHAR(255) NOT NULL,

"lastName" NVARCHAR(255) NULL,

"username" NVARCHAR(255) NOT NULL,

"mobile" BIGINT NOT NULL,

"email" NVARCHAR(255) NOT NULL,

"address" NVARCHAR(255) NOT NULL,

"registeredAt" DATETIME NOT NULL DEFAULT GETDATE()

);

ALTER TABLE

"user" ADD CONSTRAINT "user\_id\_primary" PRIMARY KEY("id");

CREATE UNIQUE INDEX "user\_username\_unique" ON

"user"("username");

CREATE UNIQUE INDEX "user\_mobile\_unique" ON

"user"("mobile");

CREATE UNIQUE INDEX "user\_email\_unique" ON

"user"("email");

ALTER TABLE

"order\_item" ADD CONSTRAINT "order\_item\_product\_id\_foreign" FOREIGN KEY("product\_id") REFERENCES "product"("id") ON DELETE NO ACTION;

ALTER TABLE

"item" ADD CONSTRAINT "item\_product\_id\_foreign" FOREIGN KEY("product\_id") REFERENCES "product"("id") ON DELETE CASCADE;

ALTER TABLE

"user" ADD CONSTRAINT "user\_supplier\_id\_foreign" FOREIGN KEY("supplier\_id") REFERENCES "supplier"("id") ON DELETE CASCADE;

ALTER TABLE

"item" ADD CONSTRAINT "item\_supplier\_id\_foreign" FOREIGN KEY("supplier\_id") REFERENCES "supplier"("id") ON DELETE SET NULL;

ALTER TABLE

"payment" ADD CONSTRAINT "payment\_user\_id\_foreign" FOREIGN KEY("user\_id") REFERENCES "user"("id") ON DELETE SET NULL;

ALTER TABLE

"product\_category" ADD CONSTRAINT "product\_category\_product\_id\_foreign" FOREIGN KEY("product\_id") REFERENCES "product"("id") ON DELETE CASCADE;

ALTER TABLE

"payment" ADD CONSTRAINT "payment\_order\_id\_foreign" FOREIGN KEY("order\_id") REFERENCES "order"("id");

ALTER TABLE

"item" ADD CONSTRAINT "item\_brand\_id\_foreign" FOREIGN KEY("brand\_id") REFERENCES "brand"("id");

ALTER TABLE

"payment" ADD CONSTRAINT "payment\_shippintaddress\_id\_foreign" FOREIGN KEY("shippintAddress\_id") REFERENCES "shippingAddress"("id") ON DELETE SET NULL;

ALTER TABLE

"order\_item" ADD CONSTRAINT "order\_item\_item\_id\_foreign" FOREIGN KEY("item\_id") REFERENCES "item"("id") ON DELETE SET NULL;

ALTER TABLE

"product\_category" ADD CONSTRAINT "product\_category\_category\_id\_foreign" FOREIGN KEY("category\_id") REFERENCES "category"("id");

ALTER TABLE

"order\_item" ADD CONSTRAINT "order\_item\_order\_id\_foreign" FOREIGN KEY("order\_id") REFERENCES "order"("id");

ALTER TABLE

"order" ADD CONSTRAINT "order\_employee\_id\_foreign" FOREIGN KEY("employee\_id") REFERENCES "employee"("id") ON DELETE SET NULL;

ALTER TABLE

"loginSession" ADD CONSTRAINT "loginsession\_employee\_id\_foreign" FOREIGN KEY("employee\_id") REFERENCES "employee"("id") ON DELETE CASCADE;

ALTER TABLE

"user" ADD CONSTRAINT "user\_employee\_id\_foreign" FOREIGN KEY("employee\_id") REFERENCES "employee"("id") ON DELETE CASCADE;

ALTER TABLE

"order" ADD CONSTRAINT "order\_user\_id\_foreign" FOREIGN KEY("user\_id") REFERENCES "user"("id") ON DELETE NO ACTION;

ALTER TABLE

"user" ADD CONSTRAINT "user\_customer\_id\_foreign" FOREIGN KEY("customer\_id") REFERENCES "customer"("id") ON DELETE CASCADE;

ALTER TABLE

"shippingAddress" ADD CONSTRAINT "shippingaddress\_customer\_id\_foreign" FOREIGN KEY("customer\_id") REFERENCES "customer"("id") ON DELETE NO ACTION;

**SCREENSHOTS:**













