# The fundamentals:

Initially the program will start a SQL connection querying the logins. This is all handled in the SqlConnector class.

Starting a Sql Connection, called at the start of the program:

 //Start the sql connnection  
        public static void startConnection()  
        {  
            string connectionString = ConfigurationManager.ConnectionStrings["farmingprogram.Properties.Settings.DbFarmProgramConnectionString"].ConnectionString; //Gets the Connection String from resources ('App.Config')  
            connection = new SqlConnection(connectionString); //Creates a new connection  
            try  
            {  
                connection.Open(); //Opens connection  
            }  
            catch (Exception ex)  
            {  
                //Any errors will leave program useless  
                //So it shuts down  
                MessageBox.Show("Error opening connection. Closing program.");  
                Environment.Exit(0);  
            }  
        }

Then it will query from the passwords and usernames. Where the passwords are encrypted by the PasswordEncryption.cs. The query string will get all login and password entries, then the SqlCommand will be executed. If the password matches the database password the login detail is set. The login detail has to be set for the next UI to open.

 //Checks if password is valid with the arguments username, password they are both inputs from the user  
        public static Boolean validPassword(String username, String password)  
        {  
            if (String.IsNullOrEmpty(username) || String.IsNullOrEmpty(password)) //If the username and password is empty then Invalid credentials would be displayed  
            {  
                MessageBox.Show("Invalid credentials.");  
                return false;  
            }  
              
            string queryString = "SELECT \* FROM Login"; //Selects all data from login  
            SqlDataReader command = new SqlCommand(queryString, getConnection()).ExecuteReader(); //Executes command  
  
            try  
            {  
                while (command.Read()) //While the command still has a output  
                {  
                    String name = command.GetString(0).ToLower(); //Get username from database  
                    String pass = command.GetString(1).ToLower(); //Get password from database  
                    int rights = command.GetInt32(2); //Get rights from database  
  
                    if (name.Equals(username.ToLower()) && pass.Equals(PasswordEncryption.Encrypt(password).ToLower())) //See if user and password match  
                    {  
                        Program.setLogin(new LoginDetail(name, pass, rights)); //Set login detail   
                        getConnection().Close();//Close connection  
                        return true; //Return true if match  
                    }  
                }  
                MessageBox.Show("Invalid username or password try again.");  
            }  
            finally  
            {  
                command.Close(); //Close command   
            }  
            return false; //Return false if no matches  
        }

If the login details are correct, the MainWindow will open unless the program is on testing mode. If the program is on testing mode.

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
  
namespace farmingprogram  
{  
    //Made by 1333187  
    static class Program  
    {  
        /// <summary>  
        /// The main entry point for the application.  
        /// </summary>  
        [STAThread]  
        static void Main()  
        {  
            Application.EnableVisualStyles();  
            Application.SetCompatibleTextRenderingDefault(false);  
  
            SqlConnector.startConnection();  
            if (!Constants.TESTING\_MODE) //If not testing then prompt login  
            {  
                Application.Run(new LoginInterface()); //Prompts login  
            }  
            else //If testing  
            {  
                loginDetail = new LoginDetail("admin", "admin", 1); //Default user and password = admin, admin  
                SqlConnector.getConnection().Close(); //Close connection left open for login  
            }  
            if (loginDetail != null) //If login detail is not null then run the program  
            {  
                Application.Run(new MainProgram()); //Run main program  
            }  
        }  
  
        private static LoginDetail loginDetail; //Login detail field  
  
        public static void setLogin(LoginDetail ld) //Login detail setter  
        {  
            loginDetail = ld;  
        }  
  
        public static LoginDetail getLoginDetail() //Login detail getter  
        {  
            return loginDetail;  
        }  
    }  
}

# Explaining the FarmingDataSet.cs

The FarmingDataSet.cs uses a series of data adapters. These adapters are assigned a specific insert, delete, update and select queries given by the Constants.cs class. The program will initialize all of these datasets using initializeCropSet(), initializeFieldSet(),initializeVehicleSet(), initializeHarestSet(),initializeCropSet(), initializeFertilizerSet() and initializStaffTest(). These initialize methods will create a new Sql adapter with the respective queries. It will also create the update parameters which will then be used for the cell row editing. The DataTables is a data source to the DataGridView forms in the application. The adapter.fill populates these tables and the ‘MainProgram.getSingleton().cropBindingSource.DataSource

 = cropDataTable;’ will display them on to the respective DataGridView. Each DataSet contains a unique add command which sets the parameters, executes the command then deletes the parameters. The ‘public static int deleteFromTable(SqlDataAdapter adapter,

String idParam, int id)’ deletes the data from the table by calling the delete query from the adapter. As the delete queries for these adapters have specific parameters for example ‘@CropID’ we would need to set them before deleting. Hence the argument ‘idParam’ will be the unique identifier for the ID in the database and will be deleted.

Here is the class:

using System;  
using System.Collections.Generic;  
using System.Data;  
using System.Data.SqlClient;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
//Created by and modified by 1333187  
namespace farmingprogram  
{  
    class FarmingDataSet  
    {  
        //Sql adapters which hold different update,insert,delete and select queries  
        public static SqlDataAdapter cropDataAdapter;   
        public static SqlDataAdapter fertilizerAdapter;  
        public static SqlDataAdapter fieldAdapter;  
        public static SqlDataAdapter containerAdapter;  
        public static SqlDataAdapter staffAdapter;  
        public static SqlDataAdapter harvestAdapter;  
        public static SqlDataAdapter vehicleAdapter;  
  
        //Data tables when the select query is read they are saved on these tables  
        public static DataTable cropDataTable;  
        public static DataTable fertilizerDataTable;  
        public static DataTable fieldDataTable;  
        public static DataTable containerDataTable;  
        public static DataTable staffDataTable;  
        public static DataTable harvestDataTable;  
        public static DataTable vehicleDataTable;  
  
        #region Crop Set  
        public static void initializeCropSet() //Initializes the crop data set  
        {  
  
            SqlConnector.getConnection().Open(); //Open connection  
            cropDataAdapter = new SqlDataAdapter(); //Attach adapter to cropDataAdapter  
  
            //Declare Adapter queries  
            cropDataAdapter.SelectCommand = new SqlCommand(Constants.CROP\_SELECTALL\_QUERY, SqlConnector.getConnection());   
            cropDataAdapter.DeleteCommand = new SqlCommand(Constants.CROP\_DELETE\_QUERY, SqlConnector.getConnection());  
            cropDataAdapter.InsertCommand = new SqlCommand(Constants.CROP\_INSERT\_QUERY, SqlConnector.getConnection());  
  
            //If the update command is just being added  
            if (cropDataAdapter.UpdateCommand == null)  
            {  
                cropDataAdapter.UpdateCommand = new SqlCommand(Constants.CROP\_UPDATE\_QUERY, SqlConnector.getConnection()); //Add the update query  
                setCropUpdateParams(); //Add the parameters for the update query  
            }  
            cropDataTable = new DataTable(); //Create a new crop data table  
            cropDataAdapter.Fill(cropDataTable); //Populate the data table  
            MainProgram.getSingleton().cropBindingSource.DataSource = cropDataTable; //Send the cropGridView to view the crop data table  
            SqlConnector.getConnection().Close(); //Close connection  
        }  
  
        public static void addCrop(Crop crop) //Add a crop  
        {  
            //Parameters which are set when query is execute "@varName", value  
            cropDataAdapter.InsertCommand.Parameters.Add(new SqlParameter("@CropName", crop.cropName));   
            cropDataAdapter.InsertCommand.Parameters.Add(new SqlParameter("@DatePlanted", crop.datePlanted));  
            cropDataAdapter.InsertCommand.Parameters.Add(new SqlParameter("@EstimatedHarvestDate", crop.estimatedHarvestDate));  
            cropDataAdapter.InsertCommand.Parameters.Add(new SqlParameter("@CropNotes", crop.cropNotes));  
            cropDataAdapter.InsertCommand.Parameters.Add(new SqlParameter("@FertilizerID", crop.fertilizerID));  
            cropDataAdapter.InsertCommand.Parameters.Add(new SqlParameter("@CropStatus", crop.cropStatus));  
            cropDataAdapter.InsertCommand.Parameters.Add(new SqlParameter("@LastDose", crop.lastDose));  
            cropDataAdapter.InsertCommand.Parameters.Add(new SqlParameter("@NextDose", crop.nextDose));  
            cropDataAdapter.InsertCommand.Parameters.Add(new SqlParameter("@DosedByStaff", crop.dosedByStaff));  
            cropDataAdapter.InsertCommand.Parameters.Add(new SqlParameter("@CropStorageType", crop.cropStorageType));  
            cropDataAdapter.InsertCommand.Parameters.Add(new SqlParameter("@CropMinMax", crop.cropMinMax));  
            cropDataAdapter.InsertCommand.Parameters.Add(new SqlParameter("@FieldID", crop.fieldID));  
  
            try  
            {  
                SqlConnector.getConnection().Open(); //Open connection  
                cropDataAdapter.InsertCommand.ExecuteNonQuery(); //Execute query  
            }  
            finally  
            {  
                cropDataAdapter.InsertCommand.Parameters.Clear(); //Clear params to avoid error  
                SqlConnector.getConnection().Close(); //Close connection  
            }  
        }  
  
        //Sets update params so when row is edited there is no need to input hardcoded values  
        public static void setCropUpdateParams()  
        {  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@CropName", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropName", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@DatePlanted", global::System.Data.SqlDbType.Date, 0, global::System.Data.ParameterDirection.Input, 0, 0, "DatePlanted", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@EstimatedHarvestDate", global::System.Data.SqlDbType.Date, 0, global::System.Data.ParameterDirection.Input, 0, 0, "EstimatedHarvestDate", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@CropNotes", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropNotes", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@FertilizerID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FertilizerID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@CropStatus", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropStatus", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@LastDose", global::System.Data.SqlDbType.Date, 0, global::System.Data.ParameterDirection.Input, 0, 0, "LastDose", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@NextDose", global::System.Data.SqlDbType.Date, 0, global::System.Data.ParameterDirection.Input, 0, 0, "NextDose", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@DosedByStaff", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "DosedByStaff", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@CropStorageType", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropStorageType", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@CropMinMax", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropMinMax", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@FieldID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FieldID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_CropID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_CropName", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropName", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_DatePlanted", global::System.Data.SqlDbType.Date, 0, global::System.Data.ParameterDirection.Input, 0, 0, "DatePlanted", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_EstimatedHarvestDate", global::System.Data.SqlDbType.Date, 0, global::System.Data.ParameterDirection.Input, 0, 0, "EstimatedHarvestDate", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@IsNull\_CropNotes", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropNotes", global::System.Data.DataRowVersion.Original, true, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_CropNotes", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropNotes", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@IsNull\_FertilizerID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FertilizerID", global::System.Data.DataRowVersion.Original, true, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_FertilizerID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FertilizerID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_CropStatus", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropStatus", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@IsNull\_LastDose", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "LastDose", global::System.Data.DataRowVersion.Original, true, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_LastDose", global::System.Data.SqlDbType.Date, 0, global::System.Data.ParameterDirection.Input, 0, 0, "LastDose", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@IsNull\_NextDose", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "NextDose", global::System.Data.DataRowVersion.Original, true, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_NextDose", global::System.Data.SqlDbType.Date, 0, global::System.Data.ParameterDirection.Input, 0, 0, "NextDose", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@IsNull\_DosedByStaff", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "DosedByStaff", global::System.Data.DataRowVersion.Original, true, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_DosedByStaff", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "DosedByStaff", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_CropStorageType", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropStorageType", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_CropMinMax", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropMinMax", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_FieldID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FieldID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            cropDataAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@CropID", global::System.Data.SqlDbType.Int, 4, global::System.Data.ParameterDirection.Input, 0, 0, "CropID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
        }  
        #endregion  
  
  
        #region Harvest set  
  
        public static void initializeHarvestSet()  //Initializes the harvest data set  
        {  
            SqlConnector.getConnection().Open(); //Open connection  
            harvestAdapter = new SqlDataAdapter(); //Attach adapter to harvestAdapter  
  
            //Declare Adapter queries  
            harvestAdapter.SelectCommand = new SqlCommand(Constants.HARVEST\_SELECTALL\_QUERY, SqlConnector.getConnection());  
            harvestAdapter.DeleteCommand = new SqlCommand(Constants.HARVEST\_DELETE\_QUERY, SqlConnector.getConnection());  
            harvestAdapter.InsertCommand = new SqlCommand(Constants.HARVEST\_INSERT\_QUERY, SqlConnector.getConnection());  
  
            //If the update command is just being added  
            if (harvestAdapter.UpdateCommand == null)  
            {  
                harvestAdapter.UpdateCommand = new SqlCommand(Constants.HARVEST\_UPDATE\_QUERY, SqlConnector.getConnection()); //Add the update query  
                setHarvestUpdateParams(); //Add the parameters for the update query  
            }  
            harvestDataTable = new DataTable(); //Create a new datatable  
            harvestAdapter.Fill(harvestDataTable); //File harvest data table  
            MainProgram.getSingleton().harvestBindingSource.DataSource = harvestDataTable; //The harvestGridView will now display the datatable  
            SqlConnector.getConnection().Close(); //Close connection  
        }  
  
        public static void addHarvest(Harvest harvest) //Adding a harvest  
        {  
            //@HarvestStartDate, @HarvestEndDate, @StaffRequired, @ContainerID, @CropID, @FieldID, @StaffID, @VehicleID, @StorageID  
            harvestAdapter.InsertCommand.Parameters.Add(new SqlParameter("@HarvestStartDate", harvest.harvestStartDate));  
            harvestAdapter.InsertCommand.Parameters.Add(new SqlParameter("@HarvestEndDate", harvest.harvestEndDate));  
            harvestAdapter.InsertCommand.Parameters.Add(new SqlParameter("@StaffRequired", harvest.staffRequired));  
            harvestAdapter.InsertCommand.Parameters.Add(new SqlParameter("@ContainerID", harvest.containerID));  
            harvestAdapter.InsertCommand.Parameters.Add(new SqlParameter("@CropID", harvest.cropID));  
            harvestAdapter.InsertCommand.Parameters.Add(new SqlParameter("@FieldID", harvest.fieldID));  
            harvestAdapter.InsertCommand.Parameters.Add(new SqlParameter("@VehicleID", harvest.vehicle));//This should be the vehicle id of the vehicle assigned to the harvest.  
            try  
            {  
                SqlConnector.getConnection().Open(); //Open sql connection  
                harvestAdapter.InsertCommand.ExecuteNonQuery(); //Execute query  
            }  
            finally  
            {  
                harvestAdapter.InsertCommand.Parameters.Clear(); //Clear params to avoid error  
                SqlConnector.getConnection().Close(); //Close connection  
            }  
        }  
  
        //Sets update params so when row is edited there is no need to input hardcoded values  
        public static void setHarvestUpdateParams()  
        {  
            harvestAdapter.UpdateCommand.CommandType = global::System.Data.CommandType.Text;  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@HarvestStartDate", global::System.Data.SqlDbType.Date, 0, global::System.Data.ParameterDirection.Input, 0, 0, "HarvestStartDate", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@HarvestEndDate", global::System.Data.SqlDbType.Date, 0, global::System.Data.ParameterDirection.Input, 0, 0, "HarvestEndDate", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@StaffRequired", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "StaffRequired", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@ContainerID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "ContainerID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@CropID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@FieldID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FieldID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@StaffID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "StaffID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@VehicleID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "VehicleID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@StorageID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "StorageID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_HarvestID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "HarvestID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_HarvestStartDate", global::System.Data.SqlDbType.Date, 0, global::System.Data.ParameterDirection.Input, 0, 0, "HarvestStartDate", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_HarvestEndDate", global::System.Data.SqlDbType.Date, 0, global::System.Data.ParameterDirection.Input, 0, 0, "HarvestEndDate", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_StaffRequired", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "StaffRequired", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_ContainerID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "ContainerID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_CropID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "CropID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_FieldID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FieldID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_StaffID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "StaffID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_VehicleID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "VehicleID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_StorageID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "StorageID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            harvestAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@HarvestID", global::System.Data.SqlDbType.Int, 4, global::System.Data.ParameterDirection.Input, 0, 0, "HarvestID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
        }  
        #endregion  
  
        #region Fertilizer set  
        public static void initializeFertilizerSet() //Initializes the fertilizer dataset  
        {  
            SqlConnector.getConnection().Open(); //Open connection  
            fertilizerAdapter = new SqlDataAdapter(); //Create a new adapter for fertilizer  
  
            //Insert queries into adapter  
            fertilizerAdapter.SelectCommand = new SqlCommand(Constants.FERTILIZER\_SELECTALL\_QUERY, SqlConnector.getConnection());  
            fertilizerAdapter.DeleteCommand = new SqlCommand(Constants.FERTILIZER\_DELETE\_QUERY, SqlConnector.getConnection());  
            fertilizerAdapter.InsertCommand = new SqlCommand(Constants.FERTILIZER\_INSERT\_QUERY, SqlConnector.getConnection());  
  
            //Update command  
            if (fertilizerAdapter.UpdateCommand == null)  
            {  
                fertilizerAdapter.UpdateCommand = new SqlCommand(Constants.FERTILIZER\_UPDATE\_QUERY, SqlConnector.getConnection()); //Add the update command  
                setFertilizerUpdateParams(); //Set the update command params  
            }  
            fertilizerDataTable = new DataTable(); //New data table for fertilizer  
            fertilizerAdapter.Fill(fertilizerDataTable); //populate data table  
            MainProgram.getSingleton().fertilizerBindingSource.DataSource = fertilizerDataTable; //Display data table on the grid  
            SqlConnector.getConnection().Close();  
        }  
  
        public static void addFertilizer(Fertilizer fertilizer)  
        {  
            fertilizerAdapter.InsertCommand.Parameters.Add(new SqlParameter("@fertName", fertilizer.fertName));  
            fertilizerAdapter.InsertCommand.Parameters.Add(new SqlParameter("@fertDose", fertilizer.fertDose));  
            fertilizerAdapter.InsertCommand.Parameters.Add(new SqlParameter("@fertNote", fertilizer.fertNote));  
            try  
            {  
                SqlConnector.getConnection().Open();  
                fertilizerAdapter.InsertCommand.ExecuteNonQuery();  
            }  
            finally  
            {  
                fertilizerAdapter.InsertCommand.Parameters.Clear();  
                SqlConnector.getConnection().Close();  
            }  
        }  
  
        //Sets update params so when row is edited there is no need to input hardcoded values  
        private static void setFertilizerUpdateParams()  
        {  
            fertilizerAdapter.UpdateCommand.CommandType = global::System.Data.CommandType.Text;  
            fertilizerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@fertName", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "fertName", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            fertilizerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@fertDose", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "fertDose", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            fertilizerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@fertNote", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "fertNote", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            fertilizerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_FertilizerID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FertilizerID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            fertilizerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_fertName", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "fertName", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            fertilizerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_fertDose", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "fertDose", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            fertilizerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_fertNote", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "fertNote", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            fertilizerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@FertilizerID", global::System.Data.SqlDbType.Int, 4, global::System.Data.ParameterDirection.Input, 0, 0, "FertilizerID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
        }  
        #endregion  
  
        #region Vehicle set  
        //Initializes vehicle set  
        public static void initializeVehicleSet()  
        {  
            SqlConnector.getConnection().Open(); //Opens connection  
            vehicleAdapter = new SqlDataAdapter(); //Binds vehicleAdapter to a new data adapter  
  
            //Sets adapter commands  
            vehicleAdapter.SelectCommand = new SqlCommand(Constants.VEHICLE\_SELECTALL\_QUERY, SqlConnector.getConnection());  
            vehicleAdapter.DeleteCommand = new SqlCommand(Constants.VEHICLE\_DELETE\_QUERY, SqlConnector.getConnection());  
            vehicleAdapter.InsertCommand = new SqlCommand(Constants.VEHICLE\_INSERT\_QUERY, SqlConnector.getConnection());  
  
            //Sets update command and parameters  
            if (vehicleAdapter.UpdateCommand != null)  
            {  
                vehicleAdapter.UpdateCommand = new SqlCommand(Constants.VEHICLE\_UPDATE\_QUERY, SqlConnector.getConnection());  
                setVehicleUpdateParams();  
            }  
  
            //populates table with database data  
            vehicleDataTable = new DataTable();  
            vehicleAdapter.Fill(vehicleDataTable);  
  
            //View the data on a grid  
            MainProgram.getSingleton().vehicleBindingSource.DataSource = vehicleDataTable;  
            SqlConnector.getConnection().Close(); //Close connection  
        }  
  
        //Add a new vehicle  
        public static void addVehicle(Vehicle vehicle)  
        {  
            //Parameters which are set when query is execute  
            vehicleAdapter.InsertCommand.Parameters.Add(new SqlParameter("@Type", vehicle.type));  
            vehicleAdapter.InsertCommand.Parameters.Add(new SqlParameter("@Make", vehicle.make));  
            vehicleAdapter.InsertCommand.Parameters.Add(new SqlParameter("@Model", vehicle.model));  
            vehicleAdapter.InsertCommand.Parameters.Add(new SqlParameter("@Description", vehicle.description));  
            vehicleAdapter.InsertCommand.Parameters.Add(new SqlParameter("@Capacity", vehicle.capacity));  
  
            try  
            {  
                SqlConnector.getConnection().Open(); //Open connection  
                vehicleAdapter.InsertCommand.ExecuteNonQuery(); //Execute query  
            }  
            finally  
            {  
                vehicleAdapter.InsertCommand.Parameters.Clear(); //Clear params to avoid error  
                SqlConnector.getConnection().Close(); //Close connection  
            }  
        }  
  
        //Adds variable parameters to update command  
        static void setVehicleUpdateParams()  
        {  
            vehicleAdapter.UpdateCommand.CommandType = global::System.Data.CommandType.Text;  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Type", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "Type", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Make", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "Make", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Model", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "Model", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Description", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "Description", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Capacity", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "Capacity", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_VehicleID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "VehicleID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_Type", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "Type", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_Make", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "Make", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_Model", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "Model", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@IsNull\_Description", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "Description", global::System.Data.DataRowVersion.Original, true, null, "", "", ""));  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_Description", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "Description", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_Capacity", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "Capacity", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            vehicleAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@VehicleID", global::System.Data.SqlDbType.Int, 4, global::System.Data.ParameterDirection.Input, 0, 0, "VehicleID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
        }  
        #endregion  
  
        //Deletes from the actual database where ID is the primary key and id param is the param string  
        public static int deleteFromTable(SqlDataAdapter adapter, String idParam, int id)   
        {  
            int returnCode = 0; //Return code for how many rows deleted  
            try  
            {  
                SqlConnector.getConnection().Open(); //Open connection  
                adapter.DeleteCommand.Parameters.Add(new SqlParameter(idParam, SqlDbType.Int)).Value = id; //Add the Param Value  
                returnCode = adapter.DeleteCommand.ExecuteNonQuery(); //execute the query and set return code  
            }  
            catch (Exception exception) //exceptions in foreign keys hence we handle then.  
            {  
                System.Windows.Forms.MessageBox.Show("Cannot delete the data, it is used on another table. Please delete this data first.");  
            }   
            finally //Clear params and close connection  
            {  
                adapter.DeleteCommand.Parameters.Clear();  
                SqlConnector.getConnection().Close();  
            }  
            return returnCode;  
        }  
  
  
  
  
        #region Field set  
        //Sets update params so when row is edited there is no need to input hardcoded values  
        private static void setFieldUpdateParams()  
        {  
            fieldAdapter.UpdateCommand.CommandType = global::System.Data.CommandType.Text;  
            fieldAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@FieldName", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FieldName", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            fieldAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@FieldStatus", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FieldStatus", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            fieldAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@FieldNotes", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FieldNotes", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            fieldAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_FieldID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FieldID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            fieldAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_FieldName", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FieldName", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            fieldAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_FieldStatus", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FieldStatus", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            fieldAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@IsNull\_FieldNotes", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FieldNotes", global::System.Data.DataRowVersion.Original, true, null, "", "", ""));  
            fieldAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_FieldNotes", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "FieldNotes", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            fieldAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@FieldID", global::System.Data.SqlDbType.Int, 4, global::System.Data.ParameterDirection.Input, 0, 0, "FieldID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
        }  
        public static void initializeFieldSet()  
        {  
            SqlConnector.getConnection().Open();  
            fieldAdapter = new SqlDataAdapter();  
            fieldAdapter.SelectCommand = new SqlCommand(Constants.FIELD\_SELECTALL\_QUERY, SqlConnector.getConnection());  
            fieldAdapter.DeleteCommand = new SqlCommand(Constants.FIELD\_DELETE\_QUERY, SqlConnector.getConnection());  
            fieldAdapter.InsertCommand = new SqlCommand(Constants.FIELD\_INSERT\_QUERY, SqlConnector.getConnection());  
  
            if (fieldAdapter.UpdateCommand == null)  
            {  
                fieldAdapter.UpdateCommand = new SqlCommand(Constants.FIELD\_UPDATE\_QUERY, SqlConnector.getConnection());  
                setFieldUpdateParams();  
            }  
            fieldDataTable = new DataTable();  
            fieldAdapter.Fill(fieldDataTable);  
            MainProgram.getSingleton().fieldBindingSource.DataSource = fieldDataTable;  
            SqlConnector.getConnection().Close();  
        }  
  
        public static void addField(Field field) //Add a field  
        {  
            //Set the param values for the query  
            fieldAdapter.InsertCommand.Parameters.Add(new SqlParameter("@FieldName", field.fieldName));   
            fieldAdapter.InsertCommand.Parameters.Add(new SqlParameter("@FieldStatus", field.fieldStatus));  
            fieldAdapter.InsertCommand.Parameters.Add(new SqlParameter("@FieldNotes", field.fieldNotes));  
  
            try  
            {  
                //Open a connection then execute the query  
                SqlConnector.getConnection().Open();  
                fieldAdapter.InsertCommand.ExecuteNonQuery();  
            }  
            finally  
            {  
  
                //Clear params and close connection  
                fieldAdapter.InsertCommand.Parameters.Clear();  
                SqlConnector.getConnection().Close();  
            }  
        }  
        #endregion  
  
  
        #region Container set  
        public static void initializeContainerSet()  
        {  
            SqlConnector.getConnection().Open();  
            containerAdapter = new SqlDataAdapter();  
            containerAdapter.SelectCommand = new SqlCommand(Constants.CONTAINER\_SELECTALL\_QUERY, SqlConnector.getConnection());  
            containerAdapter.DeleteCommand = new SqlCommand(Constants.CONTAINER\_DELETE\_QUERY, SqlConnector.getConnection());  
            containerAdapter.InsertCommand = new SqlCommand(Constants.CONTAINER\_INSERT\_QUERY, SqlConnector.getConnection());  
  
            if (containerAdapter.UpdateCommand == null)  
            {  
                containerAdapter.UpdateCommand = new SqlCommand(Constants.CONTAINER\_UPDATE\_QUERY);  
                setContainerUpdateParams();  
            }  
  
            containerDataTable = new DataTable();  
            containerAdapter.Fill(containerDataTable);  
            MainProgram.getSingleton().containerBindingSource.DataSource = containerDataTable;  
            SqlConnector.getConnection().Close();  
        }  
  
        public static void addContainer(Container container)  
        {  
            containerAdapter.InsertCommand.Parameters.Add(new SqlParameter("@contName", container.contName));  
            containerAdapter.InsertCommand.Parameters.Add(new SqlParameter("@contSize", container.contSize));  
            containerAdapter.InsertCommand.Parameters.Add(new SqlParameter("@contType", container.contType));  
            try  
            {  
                SqlConnector.getConnection().Open();  
                containerAdapter.InsertCommand.ExecuteNonQuery();  
            }  
            finally  
            {  
                containerAdapter.InsertCommand.Parameters.Clear();  
                SqlConnector.getConnection().Close();  
            }  
        }    
  
        //Sets update params so when row is edited there is no need to input hardcoded values  
        private static void setContainerUpdateParams()  
        {  
            containerAdapter.UpdateCommand.CommandType = global::System.Data.CommandType.Text;  
            containerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@contName", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "contName", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            containerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@contType", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "contType", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            containerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@contSize", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "contSize", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
            containerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_ContainerID", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "ContainerID", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            containerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_contName", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "contName", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            containerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_contType", global::System.Data.SqlDbType.NVarChar, 0, global::System.Data.ParameterDirection.Input, 0, 0, "contType", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            containerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@Original\_contSize", global::System.Data.SqlDbType.Int, 0, global::System.Data.ParameterDirection.Input, 0, 0, "contSize", global::System.Data.DataRowVersion.Original, false, null, "", "", ""));  
            containerAdapter.UpdateCommand.Parameters.Add(new global::System.Data.SqlClient.SqlParameter("@ContainerID", global::System.Data.SqlDbType.Int, 4, global::System.Data.ParameterDirection.Input, 0, 0, "ContainerID", global::System.Data.DataRowVersion.Current, false, null, "", "", ""));  
        }  
        #endregion  
  
        public static void initializeStaffSet()  
        {  
            SqlConnector.getConnection().Open();  
            staffAdapter = new SqlDataAdapter();  
            staffAdapter.SelectCommand = new SqlCommand(Constants.STAFF\_SELECTALL\_QUERY, SqlConnector.getConnection());  
            staffAdapter.DeleteCommand = new SqlCommand(Constants.STAFF\_DELETE\_QUERY, SqlConnector.getConnection());  
            staffAdapter.InsertCommand = new SqlCommand(Constants.STAFF\_INSERT\_QUERY, SqlConnector.getConnection());  
            staffDataTable = new DataTable();  
            staffAdapter.Fill(staffDataTable);  
            MainProgram.getSingleton().staffBindingSource.DataSource = staffDataTable;  
            SqlConnector.getConnection().Close();  
        }  
       
  
    }  
}

# Explaining the MainProgram.cs

The main program is the form class. It handles clicking buttons and the tab grid view. It also handles deleting, inserting and updating data on the UI layer. An instance of this object was created so that the binding source can be accessed to set by the ‘FarmingDataSet.cs’. When the data is edited in the data grid view, each data grid view has a respective method to handle the cellRowEdited. The row edit will be validated but no error handling has been added to the DataGridViewRow editing. Hence exceptions can be thrown at the user when editing a cell. The ‘adapter.Update(table);’ in ‘updateRow(BindingSource bindingSource, SqlDataAdapter adapter, DataTable table)’ updates the database with the GridView data table. The insert/add buttons for the data grid view are handled by multiple add methods and all contain the method ‘handleNullOrWhitespace

(TextBox box)’ multiple times. The objects which the database contains that cannot be null will stop the user from entering the values as null, given this method is called and the appropriate text input form. The ‘removeRow(DataGridView dgv, String idParam, int idCell,

SqlDataAdapter adapter)’ removes an object in the database. With the ‘idParam’ as the unique identifier string for the ID in the database.

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 System.Collections;  
using System.Data.SqlClient;  
  
namespace farmingprogram  
{  
    //Created by 1333187  
    public partial class MainProgram : Form  
    {  
        public MainProgram()  
        {  
            InitializeComponent();  
            programInstance = this; //Sets the program instance  
        }  
  
        static MainProgram programInstance = null; //Program instance to grab binding sources from  
  
  
        public static MainProgram getSingleton() //Singleton pattern followed 'http://www.yoda.arachsys.com/csharp/singleton.html'  
        {  
            return programInstance;  
        }  
  
        //On the load of the main program all datasets are initilialized.  
        private void MainProgram\_Load(object sender, EventArgs e)  
        {  
            try  
            {  
                //All datasets initialized  
                FarmingDataSet.initializeVehicleSet();  
                FarmingDataSet.initializeFieldSet();  
                FarmingDataSet.initializeStaffSet();  
                FarmingDataSet.initializeFertilizerSet();  
                FarmingDataSet.initializeContainerSet();  
                FarmingDataSet.initializeHarvestSet();  
                FarmingDataSet.initializeCropSet();  
            }  
            catch (Exception er) //If not show error  
            {  
                MessageBox.Show(er.ToString());  
            }  
            this.Refresh();  
        }  
  
        #region Crop Tab  
        private void removeCropButton\_Click(object sender, EventArgs e) //Remove a crop  
        {  
            removeRow(cropGridView, "@CropID", 0, FarmingDataSet.cropDataAdapter); //Remove the crop via cropId from the database and GUI  
        }  
  
        private void addCropButton\_Click(object sender, EventArgs e) //Add a crop  
        {  
            if (handleNullOrWhitespace(cropName) || handleNullOrWhitespace(cropStatus) || handleNullOrWhitespace(cropMinMax)) //Do not allow nulls of cropName or cropStatus or cropMinMax  
            {  
                return;  
            }  
            //New crop object created from user input  
            Crop crop  = new Crop(0, cropName.Text, datePlanted.Value, estimatedHarvest.Value, cropNotes.Text, Int32.Parse(fertilizerId.SelectedValue.ToString()), cropStatus.Text, lastDose.Value, nextDose.Value, Int32.Parse(dosedBy.SelectedValue.ToString()), Int32.Parse(containerStorageType.SelectedValue.ToString()), cropMinMax.Text, Int32.Parse(fieldId.SelectedValue.ToString()));  
             
            //Add the crop to database  
            FarmingDataSet.addCrop(crop);  
  
            //Reinitialize crop data set  
            FarmingDataSet.initializeCropSet();  
        }  
  
        private void cropRowEdited(object sender, DataGridViewCellEventArgs e) //Called when a row is edited  
        {  
            updateRow(cropBindingSource, FarmingDataSet.cropDataAdapter, FarmingDataSet.cropDataTable); //Update the database when a row is edited  
        }  
        #endregion  
  
        #region Fertilizer tab  
        private void removeFertilizer\_Click(object sender, EventArgs e) //Remove a fertilizer  
        {  
            removeRow(fertilizerGridView, "@FertilizerID", 0, FarmingDataSet.fertilizerAdapter); //Remove fertilizer from the database and UI  
        }  
  
        private void addFertilizer\_Click(object sender, EventArgs e) //Add a fertilizer  
        {  
            if (handleNullOrWhitespace(fertilizerName) || handleNullOrWhitespace(fertilizerNote) || handleNullOrWhitespace(fertilizerDose)) //Do not allow nulls of fertilizerName fertilizerNote and fertilizerDose  
            {  
                return;  
            }  
            Fertilizer fertilizer = new Fertilizer(0, fertilizerName.Text, fertilizerDose.Text, fertilizerNote.Text);  //Create new fertilizer object with given user input  
            FarmingDataSet.addFertilizer(fertilizer); //Add fertilizer to database  
            FarmingDataSet.initializeFertilizerSet(); //Reinit the fertilizer set  
        }  
  
        private void fertilizerRowEdited(object sender, DataGridViewCellEventArgs e) //Edited a fertilizer  
        {  
            updateRow(fertilizerBindingSource, FarmingDataSet.fertilizerAdapter, FarmingDataSet.fertilizerDataTable); //Update the row in the database  
        }  
        #endregion  
  
        #region Field Tab  
        private void addFieldButton\_Click(object sender, EventArgs e) //Add a field  
        {  
            if (handleNullOrWhitespace(fieldNameBox) || handleNullOrWhitespace(fieldStatusBox)) //fieldName and fieldStatus cannot be null  
            {  
                return;  
            }  
            Field field = new Field(0, fieldNameBox.Text, fieldStatusBox.Text, fieldNotesBox.Text); //Field object is created  
            FarmingDataSet.addField(field); //Field is then added to the database  
            FarmingDataSet.initializeFieldSet(); //Field data set is reinitialized  
        }  
  
        private void removeFieldButton\_Click(object sender, EventArgs e) //Remove a field  
        {  
            removeRow(fieldGridView, "@FieldID", 0, FarmingDataSet.fieldAdapter); //Remove field from the database  
        }  
  
        private void fieldRowEdited(object sender, DataGridViewCellEventArgs e) //Edit a field  
        {  
            updateRow(fieldBindingSource, FarmingDataSet.fieldAdapter, FarmingDataSet.fieldDataTable); //Update field and update database  
        }  
        #endregion  
          
        #region Container tab  
        private void addContainerButton\_Click(object sender, EventArgs e)  
        {  
            if (handleNullOrWhitespace(containerNameBox) || handleNullOrWhitespace(containerSizeBox) || handleNullOrWhitespace(containerTypeBox)) //containerName, containerSize, containerType cannot be null  
            {  
                return;  
            }  
  
            //Parse container size to check if it is a int  
            int containerSize;   
            Boolean isInteger = int.TryParse(containerSizeBox.Text, out containerSize);  
              
            if(!isInteger) { //If containerSize is not a integer then prompt and edit  
                MessageBox.Show("The container size field should be an integer.");  
                return;  
            }  
                  
            Container container = new Container(0, containerNameBox.Text, containerTypeBox.Text, containerSize); //Create a new container object  
            FarmingDataSet.addContainer(container); //Add the container to the database  
            FarmingDataSet.initializeContainerSet(); //Initialize the container dataset again  
        }  
  
        private void removeContainerButton\_Click(object sender, EventArgs e) //Remove container  
        {  
            removeRow(containerGridView, "@ContainerID", 0, FarmingDataSet.cropDataAdapter); //Remove the container from the database  
        }  
  
        private void containerRowEdited(object sender, DataGridViewCellEventArgs e)  
        {  
            updateRow(containerBindingSource, FarmingDataSet.containerAdapter, FarmingDataSet.containerDataTable); //Update the container from the database  
        }  
        #endregion  
  
        #region Harvest tab  
        private void addHarvestButton\_Click(object sender, EventArgs e) //Add a harvest  
        {  
            Harvest harvest = new Harvest(0, startDateBox.Value, endDateBox.Value, (int)staffOperatorComboBox.SelectedValue, (int)containerComboBox.SelectedValue, (int)cropComboBox.SelectedValue, (int)fieldComboBox.SelectedValue, (int)vehicleComboBox.SelectedValue); //A harvest object is created  
            FarmingDataSet.addHarvest(harvest); //Add a harvest  
            FarmingDataSet.initializeHarvestSet(); //Initialize the harvest dataset  
        }  
        private void harvestRowEdited(object sender, DataGridViewCellEventArgs e) //Edit the harvest row  
        {  
            updateRow(harvestBindingSource, FarmingDataSet.harvestAdapter, FarmingDataSet.harvestDataTable); //Updates changes to database  
        }  
  
        private void removeHarvestButton\_Click(object sender, EventArgs e) //Remove the harvest  
        {  
            removeRow(harvestGridView, "@HarvestID", 0, FarmingDataSet.harvestAdapter); //Removes harvest based on ID  
        }  
        #endregion  
  
        #region Vehicle tab  
        //Adds a vehicle  
        private void addVehicleButton\_Click(object sender, EventArgs e)  
        {  
            //Cannot be null variables in database  
            if (handleNullOrWhitespace(this.vehicleTypeBox) || handleNullOrWhitespace(this.vehicleMakeBox) || handleNullOrWhitespace(this.vehicleModelBox) || handleNullOrWhitespace(this.vehicleCapacityBox))  
            {  
                return;  
            }  
  
            string type = vehicleTypeBox.Text;  
            string make = vehicleMakeBox.Text;  
            string model = vehicleModelBox.Text;  
            string capacityText = vehicleCapacityBox.Text;  
  
            //Check to see if capacity is a number  
            Boolean isInt = false;  
            int capacity;  
            isInt = int.TryParse(capacityText, out capacity);  
  
            if (!isInt) //If capacity is not a number  
            {  
                MessageBox.Show("The capacity has to be a number. Try again.");  
                return;  
            }  
            Vehicle vehicle = new Vehicle(0, type, make, model, vehicleDescBox.Text, capacity);  
            FarmingDataSet.addVehicle(vehicle); //Adds vehicle to dataset  
            FarmingDataSet.initializeVehicleSet(); //Initializes vehicle again  
        }  
  
        //Removes a vehicle  
        private void removeVechicleButton\_Click(object sender, EventArgs e)  
        {  
            removeRow(vehicleGridView, "@VehicleID", 0, FarmingDataSet.vehicleAdapter); //Removes vehicle from database  
        }  
          
        //Editing vehicle Row  
        private void vehicleRowEdited(object sender, DataGridViewCellEventArgs e) //Edit the vehicle row  
        {  
            updateRow(vehicleBindingSource, FarmingDataSet.vehicleAdapter, FarmingDataSet.vehicleDataTable); //Updates changes to database  
        }  
  
        #endregion  
  
        //Updates via database any modifications to existing data  
        private void updateRow(BindingSource bindingSource, SqlDataAdapter adapter, DataTable table)   
        {  
            try  
            {  
                SqlConnector.getConnection().Open(); //Open connection  
                Validate(); //Validate edit  
                bindingSource.EndEdit(); //Finish editing  
                adapter.Update(table); //Update the existing data with new data.  
                SqlConnector.getConnection().Close(); //Close connection  
            }  
            catch (Exception ignore)  
            {  
            }  
        }  
  
        //Removes a row from DataGridView dgv, where the idParam is the parameter string, and the idCell is the id row index of the dataGridView  
        private static void removeRow(DataGridView dgv, String idParam, int idCell, SqlDataAdapter adapter)  
        {  
            Boolean notifyDelete = false; //If data was deleted this flag will turn true  
            if (dgv.SelectedRows.Count > 0) //More than 1 selected row  
            {  
                int toDelete = dgv.SelectedRows.Count; //The amount to delete  
                while (toDelete-- > 0) //Whilst there is more to delete  
                {  
                    int indexToDelete = dgv.SelectedRows[toDelete].Index; //Find index to delete  
                    if (FarmingDataSet.deleteFromTable(adapter, idParam, (int)dgv.Rows[indexToDelete].Cells[idCell].Value) == 1) //Delete from database  
                    {  
                        dgv.Rows.Remove(dgv.Rows[indexToDelete]); //Remove from GUI  
                        notifyDelete = true; //notify user  
                    }  
                }  
                if (notifyDelete) //Notify user  
                {  
                    MessageBox.Show("The data was deleted sucessfully.");  
                }  
            }  
            else //No selected rows  
            {  
                MessageBox.Show("You have nothing selected to delete");   
            }  
        }  
  
        //Checks for empty text boxes  
        public Boolean handleNullOrWhitespace(TextBox box)  
        {  
            if (String.IsNullOrWhiteSpace(box.Text)) //If the text box is empty notify user  
            {  
                MessageBox.Show(box.Name + " cannot be empty.");  
                return true; //Return true if empty  
            }  
            return false; //Return false if not empty  
        }  
    }  
}