Logbook

SID:1333187

# Entry – 16/11/2014 16:37

Added all my SID’s to the classes SqlConnector, SplashScreen, MainWindow, Crop, LoginDetail, LogIn and Program.cs.

# Entry – 16/11/2014 21:23

Fixed the remove crop button.

Changes: MainWindow.cs

1. **private** **void** removeCropButton\_Click(**object** sender, EventArgs e)
2. {
3. Crop crop = dataGridToCrop(dataGridView1.Rows[0]);
5. Clipboard.SetText(cropTableAdapter.Adapter.DeleteCommand.CommandText, TextDataFormat.Text);
6. MessageBox.Show(cropTableAdapter.Adapter.DeleteCommand.CommandText);
7. **if** (dataGridView.SelectedRows.Count > 0)
8. {
9. **int** toDelete = dataGridView.SelectedRows.Count;
10. **while** (toDelete-- > 0)
11. {
12. Crop crop = dataGridToCrop(dataGridView.SelectedRows[toDelete]);
13. farmingDataSet.Crop.RemoveCropRow(farmingDataSet.Crop.FindByCropID(crop.cropId));
14. cropTableAdapter.Delete(crop.cropId, crop.cropName, crop.datePlanted, crop.estimatedHarvestDate, crop.cropNotes, crop.fertilizerID, crop.cropStatus, crop.lastDose, crop.nextDose, crop.dosedByStaff, crop.cropStorageType, crop.cropMinMax, crop.fieldID);
15. cropTableAdapter.Update(farmingDataSet.Crop);
16. }
17. }
18. **else**
19. {
20. MessageBox.Show("You have nothing selected to delete");
21. }
22. }

Added fertilizer and field loading:

Changes: MainWindow.cs

1. **private** **void** MainProgram\_Load(**object** sender, EventArgs e)
2. {
3. *// TODO: This line of code loads data into the 'farmingDataSet.Field' table. You can move, or remove it, as needed.*
4. **this**.fieldTableAdapter.Fill(**this**.farmingDataSet.Field);
5. *// TODO: This line of code loads data into the 'farmingDataSet.Staff' table. You can move, or remove it, as needed.*
6. **this**.staffTableAdapter.Fill(**this**.farmingDataSet.Staff);
7. *// TODO: This line of code loads data into the 'farmingDataSet.Fertilizer' table. You can move, or remove it, as needed.*
8. **this**.fertilizerTableAdapter.Fill(**this**.farmingDataSet.Fertilizer);
9. *// TODO: This line of code loads data into the 'farmingDataSet.Crop' table. You can move, or remove it, as needed.*
10. **this**.cropTableAdapter.Fill(**this**.farmingDataSet.Crop);
12. }

# Entry 17/11/2014 16:43

Adding crops to the UI will now work.

Changes:

MainWindow.cs

1. **private** **void** addCropButton\_Click(**object** sender, EventArgs e)
2. {
3. cropTableAdapter.Insert(cropName.Text, datePlanted.**Value**, estimatedHarvest.**Value**, cropNotes.Text, Int32.Parse(fertilizerId.ValueMember), cropStatus.Text, lastDose.**Value**, nextDose.**Value**, Int32.Parse(dosedBy.ValueMember), Int32.Parse(containerStorageType.ValueMember), cropMinMax.Text, Int32.Parse(fieldId.ValueMember));
4. **this**.cropTableAdapter.Fill(**this**.farmingDataSet.Crop);
5. dataGridView.Refresh();
6. }

Fixed the login bug where you could close the login and access the program

Changes:

MainWindow.cs

1. SqlConnector.startConnection();
2. Application.Run(new LoginInterface());
3. Application.Run(new MainProgram());
4. **if** (loginDetail != **null**)
5. {
6. Application.Run(new MainProgram());
7. }

# Entry: 23/11/2014 – 18:33

Created the farming datasets manually and added all methods to them.

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Data;
4. **using** System.Data.SqlClient;
5. **using** System.Linq;
6. **using** System.Text;
7. **using** System.Threading.Tasks;
9. *//Created by and modified by 1333187*
10. **namespace** farmingprogram
11. {
12. **class** FarmingDataSet
13. {
14. *//Sql adapters which hold different update,insert,delete and select queries*
15. **public** **static** SqlDataAdapter cropDataAdapter;
16. **public** **static** SqlDataAdapter fertilizerAdapter;
17. **public** **static** SqlDataAdapter fieldAdapter;
18. **public** **static** SqlDataAdapter containerAdapter;
19. **public** **static** SqlDataAdapter staffAdapter;
20. **public** **static** SqlDataAdapter harvestAdapter;
21. **public** **static** SqlDataAdapter vehicleAdapter;
23. *//Data tables when the select query is read they are saved on these tables*
24. **public** **static** DataTable cropDataTable;
25. **public** **static** DataTable fertilizerDataTable;
26. **public** **static** DataTable fieldDataTable;
27. **public** **static** DataTable containerDataTable;
28. **public** **static** DataTable staffDataTable;
29. **public** **static** DataTable harvestDataTable;
30. **public** **static** DataTable vehicleDataTable;
32. #region Crop Set
33. **public** **static** **void** initializeCropSet() *//Initializes the crop data set*
34. {
36. SqlConnector.getConnection().Open(); *//Open connection*
37. cropDataAdapter = new SqlDataAdapter(); *//Attach adapter to cropDataAdapter*
39. *//Declare Adapter queries*
40. cropDataAdapter.SelectCommand = new SqlCommand(Constants.CROP\_SELECTALL\_QUERY, SqlConnector.getConnection());
41. cropDataAdapter.DeleteCommand = new SqlCommand(Constants.CROP\_DELETE\_QUERY, SqlConnector.getConnection());
42. cropDataAdapter.InsertCommand = new SqlCommand(Constants.CROP\_INSERT\_QUERY, SqlConnector.getConnection());
44. *//If the update command is just being added*
45. **if** (cropDataAdapter.UpdateCommand == **null**)
46. {
47. cropDataAdapter.UpdateCommand = new SqlCommand(Constants.CROP\_UPDATE\_QUERY, SqlConnector.getConnection()); *//Add the update query*
48. setCropUpdateParams(); *//Add the parameters for the update query*
49. }
50. cropDataTable = new DataTable(); *//Create a new crop data table*
51. cropDataAdapter.Fill(cropDataTable); *//Populate the data table*
52. MainProgram.getSingleton().cropBindingSource.DataSource = cropDataTable; *//Send the cropGridView to view the crop data table*
53. SqlConnector.getConnection().Close(); *//Close connection*
54. }
56. **public** **static** **void** addCrop(Crop crop) *//Add a crop*
57. {
58. *//Parameters which are set when query is execute "@varName", value*
59. cropDataAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@CropName", crop.cropName));
60. cropDataAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@DatePlanted", crop.datePlanted));
61. cropDataAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@EstimatedHarvestDate", crop.estimatedHarvestDate));
62. cropDataAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@CropNotes", crop.cropNotes));
63. cropDataAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@FertilizerID", crop.fertilizerID));
64. cropDataAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@CropStatus", crop.cropStatus));
65. cropDataAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@LastDose", crop.lastDose));
66. cropDataAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@NextDose", crop.nextDose));
67. cropDataAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@DosedByStaff", crop.dosedByStaff));
68. cropDataAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@CropStorageType", crop.cropStorageType));
69. cropDataAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@CropMinMax", crop.cropMinMax));
70. cropDataAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@FieldID", crop.fieldID));
72. **try**
73. {
74. SqlConnector.getConnection().Open(); *//Open connection*
75. cropDataAdapter.InsertCommand.ExecuteNonQuery(); *//Execute query*
76. }
77. **finally**
78. {
79. cropDataAdapter.InsertCommand.Parameters.Clear(); *//Clear params to avoid error*
80. SqlConnector.getConnection().Close(); *//Close connection*
81. }
82. }
84. *//Sets update params so when row is edited there is no need to input hardcoded values*
85. **public** **static** **void** setCropUpdateParams()
86. {
87. 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**, "", "", ""));
88. 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**, "", "", ""));
89. 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**, "", "", ""));
90. 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**, "", "", ""));
91. 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**, "", "", ""));
92. 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**, "", "", ""));
93. 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**, "", "", ""));
94. 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**, "", "", ""));
95. 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**, "", "", ""));
96. 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**, "", "", ""));
97. 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**, "", "", ""));
98. 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**, "", "", ""));
99. 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**, "", "", ""));
100. 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**, "", "", ""));
101. 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**, "", "", ""));
102. 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**, "", "", ""));
103. 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**, "", "", ""));
104. 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**, "", "", ""));
105. 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**, "", "", ""));
106. 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**, "", "", ""));
107. 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**, "", "", ""));
108. 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**, "", "", ""));
109. 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**, "", "", ""));
110. 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**, "", "", ""));
111. 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**, "", "", ""));
112. 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**, "", "", ""));
113. 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**, "", "", ""));
114. 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**, "", "", ""));
115. 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**, "", "", ""));
116. 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**, "", "", ""));
117. 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**, "", "", ""));
118. }
119. #endregion

122. #region Harvest set
124. **public** **static** **void** initializeHarvestSet()  *//Initializes the harvest data set*
125. {
126. SqlConnector.getConnection().Open(); *//Open connection*
127. harvestAdapter = new SqlDataAdapter(); *//Attach adapter to harvestAdapter*
129. *//Declare Adapter queries*
130. harvestAdapter.SelectCommand = new SqlCommand(Constants.HARVEST\_SELECTALL\_QUERY, SqlConnector.getConnection());
131. harvestAdapter.DeleteCommand = new SqlCommand(Constants.HARVEST\_DELETE\_QUERY, SqlConnector.getConnection());
132. harvestAdapter.InsertCommand = new SqlCommand(Constants.HARVEST\_INSERT\_QUERY, SqlConnector.getConnection());
134. *//If the update command is just being added*
135. **if** (harvestAdapter.UpdateCommand == **null**)
136. {
137. harvestAdapter.UpdateCommand = new SqlCommand(Constants.HARVEST\_UPDATE\_QUERY, SqlConnector.getConnection()); *//Add the update query*
138. setHarvestUpdateParams(); *//Add the parameters for the update query*
139. }
140. harvestDataTable = new DataTable(); *//Create a new datatable*
141. harvestAdapter.Fill(harvestDataTable); *//File harvest data table*
142. MainProgram.getSingleton().harvestBindingSource.DataSource = harvestDataTable; *//The harvestGridView will now display the datatable*
143. SqlConnector.getConnection().Close(); *//Close connection*
144. }
146. **public** **static** **void** addHarvest(Harvest harvest) *//Adding a harvest*
147. {
148. *//@HarvestStartDate, @HarvestEndDate, @StaffRequired, @ContainerID, @CropID, @FieldID, @StaffID, @VehicleID, @StorageID*
149. harvestAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@HarvestStartDate", harvest.harvestStartDate));
150. harvestAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@HarvestEndDate", harvest.harvestEndDate));
151. harvestAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@StaffRequired", harvest.staffRequired));
152. harvestAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@ContainerID", harvest.containerID));
153. harvestAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@CropID", harvest.cropID));
154. harvestAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@FieldID", harvest.fieldID));
155. harvestAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@VehicleID", harvest.vehicle));*//This should be the vehicle id of the vehicle assigned to the harvest.*
156. **try**
157. {
158. SqlConnector.getConnection().Open(); *//Open sql connection*
159. harvestAdapter.InsertCommand.ExecuteNonQuery(); *//Execute query*
160. }
161. **finally**
162. {
163. harvestAdapter.InsertCommand.Parameters.Clear(); *//Clear params to avoid error*
164. SqlConnector.getConnection().Close(); *//Close connection*
165. }
166. }
168. *//Sets update params so when row is edited there is no need to input hardcoded values*
169. **public** **static** **void** setHarvestUpdateParams()
170. {
171. harvestAdapter.UpdateCommand.CommandType = global::System.Data.CommandType.Text;
172. 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**, "", "", ""));
173. 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**, "", "", ""));
174. 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**, "", "", ""));
175. 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**, "", "", ""));
176. 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**, "", "", ""));
177. 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**, "", "", ""));
178. 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**, "", "", ""));
179. 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**, "", "", ""));
180. 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**, "", "", ""));
181. 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**, "", "", ""));
182. 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**, "", "", ""));
183. 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**, "", "", ""));
184. 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**, "", "", ""));
185. 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**, "", "", ""));
186. 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**, "", "", ""));
187. 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**, "", "", ""));
188. 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**, "", "", ""));
189. 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**, "", "", ""));
190. 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**, "", "", ""));
191. 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**, "", "", ""));
192. }
193. #endregion
195. #region Fertilizer set
196. **public** **static** **void** initializeFertilizerSet() *//Initializes the fertilizer dataset*
197. {
198. SqlConnector.getConnection().Open(); *//Open connection*
199. fertilizerAdapter = new SqlDataAdapter(); *//Create a new adapter for fertilizer*
201. *//Insert queries into adapter*
202. fertilizerAdapter.SelectCommand = new SqlCommand(Constants.FERTILIZER\_SELECTALL\_QUERY, SqlConnector.getConnection());
203. fertilizerAdapter.DeleteCommand = new SqlCommand(Constants.FERTILIZER\_DELETE\_QUERY, SqlConnector.getConnection());
204. fertilizerAdapter.InsertCommand = new SqlCommand(Constants.FERTILIZER\_INSERT\_QUERY, SqlConnector.getConnection());
206. *//Update command*
207. **if** (fertilizerAdapter.UpdateCommand == **null**)
208. {
209. fertilizerAdapter.UpdateCommand = new SqlCommand(Constants.FERTILIZER\_UPDATE\_QUERY, SqlConnector.getConnection()); *//Add the update command*
210. setFertilizerUpdateParams(); *//Set the update command params*
211. }
212. fertilizerDataTable = new DataTable(); *//New data table for fertilizer*
213. fertilizerAdapter.Fill(fertilizerDataTable); *//populate data table*
214. MainProgram.getSingleton().fertilizerBindingSource.DataSource = fertilizerDataTable; *//Display data table on the grid*
215. SqlConnector.getConnection().Close();
216. }
218. **public** **static** **void** addFertilizer(Fertilizer fertilizer)
219. {
220. fertilizerAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@fertName", fertilizer.fertName));
221. fertilizerAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@fertDose", fertilizer.fertDose));
222. fertilizerAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@fertNote", fertilizer.fertNote));
223. **try**
224. {
225. SqlConnector.getConnection().Open();
226. fertilizerAdapter.InsertCommand.ExecuteNonQuery();
227. }
228. **finally**
229. {
230. fertilizerAdapter.InsertCommand.Parameters.Clear();
231. SqlConnector.getConnection().Close();
232. }
233. }
235. *//Sets update params so when row is edited there is no need to input hardcoded values*
236. **private** **static** **void** setFertilizerUpdateParams()
237. {
238. fertilizerAdapter.UpdateCommand.CommandType = global::System.Data.CommandType.Text;
239. 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**, "", "", ""));
240. 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**, "", "", ""));
241. 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**, "", "", ""));
242. 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**, "", "", ""));
243. 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**, "", "", ""));
244. 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**, "", "", ""));
245. 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**, "", "", ""));
246. 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**, "", "", ""));
247. }
248. #endregion
250. #region Vehicle set
251. *//Initializes vehicle set*
252. **public** **static** **void** initializeVehicleSet()
253. {
254. SqlConnector.getConnection().Open(); *//Opens connection*
255. vehicleAdapter = new SqlDataAdapter(); *//Binds vehicleAdapter to a new data adapter*
257. *//Sets adapter commands*
258. vehicleAdapter.SelectCommand = new SqlCommand(Constants.VEHICLE\_SELECTALL\_QUERY, SqlConnector.getConnection());
259. vehicleAdapter.DeleteCommand = new SqlCommand(Constants.VEHICLE\_DELETE\_QUERY, SqlConnector.getConnection());
260. vehicleAdapter.InsertCommand = new SqlCommand(Constants.VEHICLE\_INSERT\_QUERY, SqlConnector.getConnection());
262. *//Sets update command and parameters*
263. **if** (vehicleAdapter.UpdateCommand != **null**)
264. {
265. vehicleAdapter.UpdateCommand = new SqlCommand(Constants.VEHICLE\_UPDATE\_QUERY, SqlConnector.getConnection());
266. setVehicleUpdateParams();
267. }
269. *//populates table with database data*
270. vehicleDataTable = new DataTable();
271. vehicleAdapter.Fill(vehicleDataTable);
273. *//View the data on a grid*
274. MainProgram.getSingleton().vehicleBindingSource.DataSource = vehicleDataTable;
275. SqlConnector.getConnection().Close(); *//Close connection*
276. }
278. *//Add a new vehicle*
279. **public** **static** **void** addVehicle(Vehicle vehicle)
280. {
281. *//Parameters which are set when query is execute*
282. vehicleAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@Type", vehicle.type));
283. vehicleAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@Make", vehicle.make));
284. vehicleAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@Model", vehicle.model));
285. vehicleAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@Description", vehicle.description));
286. vehicleAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@Capacity", vehicle.capacity));
288. **try**
289. {
290. SqlConnector.getConnection().Open(); *//Open connection*
291. vehicleAdapter.InsertCommand.ExecuteNonQuery(); *//Execute query*
292. }
293. **finally**
294. {
295. vehicleAdapter.InsertCommand.Parameters.Clear(); *//Clear params to avoid error*
296. SqlConnector.getConnection().Close(); *//Close connection*
297. }
298. }
300. *//Adds variable parameters to update command*
301. **static** **void** setVehicleUpdateParams()
302. {
303. vehicleAdapter.UpdateCommand.CommandType = global::System.Data.CommandType.Text;
304. 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**, "", "", ""));
305. 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**, "", "", ""));
306. 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**, "", "", ""));
307. 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**, "", "", ""));
308. 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**, "", "", ""));
309. 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**, "", "", ""));
310. 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**, "", "", ""));
311. 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**, "", "", ""));
312. 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**, "", "", ""));
313. 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**, "", "", ""));
314. 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**, "", "", ""));
315. 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**, "", "", ""));
316. 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**, "", "", ""));
317. }
318. #endregion
320. *//Deletes from the actual database where ID is the primary key and id param is the param string*
321. **public** **static** **int** deleteFromTable(SqlDataAdapter adapter, **String** idParam, **int** id)
322. {
323. **int** returnCode = 0; *//Return code for how many rows deleted*
324. **try**
325. {
326. SqlConnector.getConnection().Open(); *//Open connection*
327. adapter.DeleteCommand.Parameters.**Add**(new SqlParameter(idParam, SqlDbType.**Int**)).**Value** = id; *//Add the Param Value*
328. returnCode = adapter.DeleteCommand.ExecuteNonQuery(); *//execute the query and set return code*
329. }
330. **catch** (Exception exception) *//exceptions in foreign keys hence we handle then.*
331. {
332. System.Windows.Forms.MessageBox.Show("Cannot delete the data, it is used on another table. Please delete this data first.");
333. }
334. **finally** *//Clear params and close connection*
335. {
336. adapter.DeleteCommand.Parameters.Clear();
337. SqlConnector.getConnection().Close();
338. }
339. **return** returnCode;
340. }



345. #region Field set
346. *//Sets update params so when row is edited there is no need to input hardcoded values*
347. **private** **static** **void** setFieldUpdateParams()
348. {
349. fieldAdapter.UpdateCommand.CommandType = global::System.Data.CommandType.Text;
350. 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**, "", "", ""));
351. 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**, "", "", ""));
352. 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**, "", "", ""));
353. 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**, "", "", ""));
354. 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**, "", "", ""));
355. 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**, "", "", ""));
356. 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**, "", "", ""));
357. 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**, "", "", ""));
358. 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**, "", "", ""));
359. }
360. **public** **static** **void** initializeFieldSet()
361. {
362. SqlConnector.getConnection().Open();
363. fieldAdapter = new SqlDataAdapter();
364. fieldAdapter.SelectCommand = new SqlCommand(Constants.FIELD\_SELECTALL\_QUERY, SqlConnector.getConnection());
365. fieldAdapter.DeleteCommand = new SqlCommand(Constants.FIELD\_DELETE\_QUERY, SqlConnector.getConnection());
366. fieldAdapter.InsertCommand = new SqlCommand(Constants.FIELD\_INSERT\_QUERY, SqlConnector.getConnection());
368. **if** (fieldAdapter.UpdateCommand == **null**)
369. {
370. fieldAdapter.UpdateCommand = new SqlCommand(Constants.FIELD\_UPDATE\_QUERY, SqlConnector.getConnection());
371. setFieldUpdateParams();
372. }
373. fieldDataTable = new DataTable();
374. fieldAdapter.Fill(fieldDataTable);
375. MainProgram.getSingleton().fieldBindingSource.DataSource = fieldDataTable;
376. SqlConnector.getConnection().Close();
377. }
379. **public** **static** **void** addField(Field field) *//Add a field*
380. {
381. *//Set the param values for the query*
382. fieldAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@FieldName", field.fieldName));
383. fieldAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@FieldStatus", field.fieldStatus));
384. fieldAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@FieldNotes", field.fieldNotes));
386. **try**
387. {
388. *//Open a connection then execute the query*
389. SqlConnector.getConnection().Open();
390. fieldAdapter.InsertCommand.ExecuteNonQuery();
391. }
392. **finally**
393. {
395. *//Clear params and close connection*
396. fieldAdapter.InsertCommand.Parameters.Clear();
397. SqlConnector.getConnection().Close();
398. }
399. }
400. #endregion

403. #region Container set
404. **public** **static** **void** initializeContainerSet()
405. {
406. SqlConnector.getConnection().Open();
407. containerAdapter = new SqlDataAdapter();
408. containerAdapter.SelectCommand = new SqlCommand(Constants.CONTAINER\_SELECTALL\_QUERY, SqlConnector.getConnection());
409. containerAdapter.DeleteCommand = new SqlCommand(Constants.CONTAINER\_DELETE\_QUERY, SqlConnector.getConnection());
410. containerAdapter.InsertCommand = new SqlCommand(Constants.CONTAINER\_INSERT\_QUERY, SqlConnector.getConnection());
412. **if** (containerAdapter.UpdateCommand == **null**)
413. {
414. containerAdapter.UpdateCommand = new SqlCommand(Constants.CONTAINER\_UPDATE\_QUERY);
415. setContainerUpdateParams();
416. }
418. containerDataTable = new DataTable();
419. containerAdapter.Fill(containerDataTable);
420. MainProgram.getSingleton().containerBindingSource.DataSource = containerDataTable;
421. SqlConnector.getConnection().Close();
422. }
424. **public** **static** **void** addContainer(Container container)
425. {
426. containerAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@contName", container.contName));
427. containerAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@contSize", container.contSize));
428. containerAdapter.InsertCommand.Parameters.**Add**(new SqlParameter("@contType", container.contType));
429. **try**
430. {
431. SqlConnector.getConnection().Open();
432. containerAdapter.InsertCommand.ExecuteNonQuery();
433. }
434. **finally**
435. {
436. containerAdapter.InsertCommand.Parameters.Clear();
437. SqlConnector.getConnection().Close();
438. }
439. }
441. *//Sets update params so when row is edited there is no need to input hardcoded values*
442. **private** **static** **void** setContainerUpdateParams()
443. {
444. containerAdapter.UpdateCommand.CommandType = global::System.Data.CommandType.Text;
445. 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**, "", "", ""));
446. 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**, "", "", ""));
447. 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**, "", "", ""));
448. 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**, "", "", ""));
449. 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**, "", "", ""));
450. 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**, "", "", ""));
451. 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**, "", "", ""));
452. 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**, "", "", ""));
453. }
454. #endregion
456. **public** **static** **void** initializeStaffSet()
457. {
458. SqlConnector.getConnection().Open();
459. staffAdapter = new SqlDataAdapter();
460. staffAdapter.SelectCommand = new SqlCommand(Constants.STAFF\_SELECTALL\_QUERY, SqlConnector.getConnection());
461. staffAdapter.DeleteCommand = new SqlCommand(Constants.STAFF\_DELETE\_QUERY, SqlConnector.getConnection());
462. staffAdapter.InsertCommand = new SqlCommand(Constants.STAFF\_INSERT\_QUERY, SqlConnector.getConnection());
463. staffDataTable = new DataTable();
464. staffAdapter.Fill(staffDataTable);
465. MainProgram.getSingleton().staffBindingSource.DataSource = staffDataTable;
466. SqlConnector.getConnection().Close();
467. }

470. }
471. }

# Entry - 23/11/2014 17:11

Fixed main design and added all views except staff. Integrated buttons and cells and other items via code.

MainProgram.cs

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.ComponentModel;
4. **using** System.Data;
5. **using** System.Drawing;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.Threading.Tasks;
9. **using** System.Windows.Forms;
10. **using** System.Collections;
11. **using** System.Data.SqlClient;
13. **namespace** farmingprogram
14. {
15. *//Created by 1333187*
16. **public** **partial** **class** MainProgram : Form
17. {
18. **public** MainProgram()
19. {
20. InitializeComponent();
21. programInstance = **this**; *//Sets the program instance*
22. }
24. **static** MainProgram programInstance = **null**; *//Program instance to grab binding sources from*

27. **public** **static** MainProgram getSingleton() *//Singleton pattern followed 'http://www.yoda.arachsys.com/csharp/singleton.html'*
28. {
29. **return** programInstance;
30. }
32. *//On the load of the main program all datasets are initilialized.*
33. **private** **void** MainProgram\_Load(**object** sender, EventArgs e)
34. {
35. **try**
36. {
37. *//All datasets initialized*
38. FarmingDataSet.initializeVehicleSet();
39. FarmingDataSet.initializeFieldSet();
40. FarmingDataSet.initializeStaffSet();
41. FarmingDataSet.initializeFertilizerSet();
42. FarmingDataSet.initializeContainerSet();
43. FarmingDataSet.initializeHarvestSet();
44. FarmingDataSet.initializeCropSet();
45. }
46. **catch** (Exception er) *//If not show error*
47. {
48. MessageBox.Show(er.ToString());
49. }
50. **this**.Refresh();
51. }
53. #region Crop Tab
54. **private** **void** removeCropButton\_Click(**object** sender, EventArgs e) *//Remove a crop*
55. {
56. removeRow(cropGridView, "@CropID", 0, FarmingDataSet.cropDataAdapter); *//Remove the crop via cropId from the database and GUI*
57. }
59. **private** **void** addCropButton\_Click(**object** sender, EventArgs e) *//Add a crop*
60. {
61. **if** (handleNullOrWhitespace(cropName) || handleNullOrWhitespace(cropStatus) || handleNullOrWhitespace(cropMinMax)) *//Do not allow nulls of cropName or cropStatus or cropMinMax*
62. {
63. **return**;
64. }
65. *//New crop object created from user input*
66. 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()));
68. *//Add the crop to database*
69. FarmingDataSet.addCrop(crop);
71. *//Reinitialize crop data set*
72. FarmingDataSet.initializeCropSet();
73. }
75. **private** **void** cropRowEdited(**object** sender, DataGridViewCellEventArgs e) *//Called when a row is edited*
76. {
77. updateRow(cropBindingSource, FarmingDataSet.cropDataAdapter, FarmingDataSet.cropDataTable); *//Update the database when a row is edited*
78. }
79. #endregion
81. #region Fertilizer tab
82. **private** **void** removeFertilizer\_Click(**object** sender, EventArgs e) *//Remove a fertilizer*
83. {
84. removeRow(fertilizerGridView, "@FertilizerID", 0, FarmingDataSet.fertilizerAdapter); *//Remove fertilizer from the database and UI*
85. }
87. **private** **void** addFertilizer\_Click(**object** sender, EventArgs e) *//Add a fertilizer*
88. {
89. **if** (handleNullOrWhitespace(fertilizerName) || handleNullOrWhitespace(fertilizerNote) || handleNullOrWhitespace(fertilizerDose))*//Do not allow nulls of fertilizerName fertilizerNote and fertilizerDose*
90. {
91. **return**;
92. }
93. Fertilizer fertilizer = new Fertilizer(0, fertilizerName.Text, fertilizerDose.Text, fertilizerNote.Text);  *//Create new fertilizer object with given user input*
94. FarmingDataSet.addFertilizer(fertilizer); *//Add fertilizer to database*
95. FarmingDataSet.initializeFertilizerSet(); *//Reinit the fertilizer set*
96. }
98. **private** **void** fertilizerRowEdited(**object** sender, DataGridViewCellEventArgs e) *//Edited a fertilizer*
99. {
100. updateRow(fertilizerBindingSource, FarmingDataSet.fertilizerAdapter, FarmingDataSet.fertilizerDataTable); *//Update the row in the database*
101. }
102. #endregion
104. #region Field Tab
105. **private** **void** addFieldButton\_Click(**object** sender, EventArgs e) *//Add a field*
106. {
107. **if** (handleNullOrWhitespace(fieldNameBox) || handleNullOrWhitespace(fieldStatusBox)) *//fieldName and fieldStatus cannot be null*
108. {
109. **return**;
110. }
111. Field field = new Field(0, fieldNameBox.Text, fieldStatusBox.Text, fieldNotesBox.Text); *//Field object is created*
112. FarmingDataSet.addField(field); *//Field is then added to the database*
113. FarmingDataSet.initializeFieldSet(); *//Field data set is reinitialized*
114. }
116. **private** **void** removeFieldButton\_Click(**object** sender, EventArgs e) *//Remove a field*
117. {
118. removeRow(fieldGridView, "@FieldID", 0, FarmingDataSet.fieldAdapter); *//Remove field from the database*
119. }
121. **private** **void** fieldRowEdited(**object** sender, DataGridViewCellEventArgs e) *//Edit a field*
122. {
123. updateRow(fieldBindingSource, FarmingDataSet.fieldAdapter, FarmingDataSet.fieldDataTable); *//Update field and update database*
124. }
125. #endregion
127. #region Container tab
128. **private** **void** addContainerButton\_Click(**object** sender, EventArgs e)
129. {
130. **if** (handleNullOrWhitespace(containerNameBox) || handleNullOrWhitespace(containerSizeBox) ||handleNullOrWhitespace(containerTypeBox)) *//containerName, containerSize, containerType cannot be null*
131. {
132. **return**;
133. }
135. *//Parse container size to check if it is a int*
136. **int** containerSize;
137. Boolean isInteger = **int**.TryParse(containerSizeBox.Text, **out** containerSize);
139. **if**(!isInteger) { *//If containerSize is not a integer then prompt and edit*
140. MessageBox.Show("The container size field should be an integer.");
141. **return**;
142. }
144. Container container = new Container(0, containerNameBox.Text, containerTypeBox.Text, containerSize); *//Create a new container object*
145. FarmingDataSet.addContainer(container); *//Add the container to the database*
146. FarmingDataSet.initializeContainerSet(); *//Initialize the container dataset again*
147. }
149. **private** **void** removeContainerButton\_Click(**object** sender, EventArgs e) *//Remove container*
150. {
151. removeRow(containerGridView, "@ContainerID", 0, FarmingDataSet.cropDataAdapter); *//Remove the container from the database*
152. }
154. **private** **void** containerRowEdited(**object** sender, DataGridViewCellEventArgs e)
155. {
156. updateRow(containerBindingSource, FarmingDataSet.containerAdapter, FarmingDataSet.containerDataTable); *//Update the container from the database*
157. }
158. #endregion
160. #region Harvest tab
161. **private** **void** addHarvestButton\_Click(**object** sender, EventArgs e) *//Add a harvest*
162. {
163. 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*
164. FarmingDataSet.addHarvest(harvest); *//Add a harvest*
165. FarmingDataSet.initializeHarvestSet(); *//Initialize the harvest dataset*
166. }
167. **private** **void** harvestRowEdited(**object** sender, DataGridViewCellEventArgs e) *//Edit the harvest row*
168. {
169. updateRow(harvestBindingSource, FarmingDataSet.harvestAdapter, FarmingDataSet.harvestDataTable); *//Updates changes to database*
170. }
172. **private** **void** removeHarvestButton\_Click(**object** sender, EventArgs e) *//Remove the harvest*
173. {
174. removeRow(harvestGridView, "@HarvestID", 0, FarmingDataSet.harvestAdapter); *//Removes harvest based on ID*
175. }
176. #endregion
178. #region Vehicle tab
179. *//Adds a vehicle*
180. **private** **void** addVehicleButton\_Click(**object** sender, EventArgs e)
181. {
182. *//Cannot be null variables in database*
183. **if** (handleNullOrWhitespace(**this**.vehicleTypeBox) || handleNullOrWhitespace(**this**.vehicleMakeBox) ||handleNullOrWhitespace(**this**.vehicleModelBox) || handleNullOrWhitespace(**this**.vehicleCapacityBox))
184. {
185. **return**;
186. }
188. **string** type = vehicleTypeBox.Text;
189. **string** make = vehicleMakeBox.Text;
190. **string** model = vehicleModelBox.Text;
191. **string** capacityText = vehicleCapacityBox.Text;
193. *//Check to see if capacity is a number*
194. Boolean isInt = **false**;
195. **int** capacity;
196. isInt = **int**.TryParse(capacityText, **out** capacity);
198. **if** (!isInt) *//If capacity is not a number*
199. {
200. MessageBox.Show("The capacity has to be a number. Try again.");
201. **return**;
202. }
203. Vehicle vehicle = new Vehicle(0, type, make, model, vehicleDescBox.Text, capacity);
204. FarmingDataSet.addVehicle(vehicle); *//Adds vehicle to dataset*
205. FarmingDataSet.initializeVehicleSet(); *//Initializes vehicle again*
206. }
208. *//Removes a vehicle*
209. **private** **void** removeVechicleButton\_Click(**object** sender, EventArgs e)
210. {
211. removeRow(vehicleGridView, "@VehicleID", 0, FarmingDataSet.vehicleAdapter); *//Removes vehicle from database*
212. }
214. *//Editing vehicle Row*
215. **private** **void** vehicleRowEdited(**object** sender, DataGridViewCellEventArgs e) *//Edit the vehicle row*
216. {
217. updateRow(vehicleBindingSource, FarmingDataSet.vehicleAdapter, FarmingDataSet.vehicleDataTable); *//Updates changes to database*
218. }
220. #endregion
222. *//Updates via database any modifications to existing data*
223. **private** **void** updateRow(BindingSource bindingSource, SqlDataAdapter adapter, DataTable table)
224. {
225. **try**
226. {
227. SqlConnector.getConnection().Open(); *//Open connection*
228. Validate(); *//Validate edit*
229. bindingSource.EndEdit(); *//Finish editing*
230. adapter.Update(table); *//Update the existing data with new data.*
231. SqlConnector.getConnection().Close(); *//Close connection*
232. }
233. **catch** (Exception ignore)
234. {
235. }
236. }
238. *//Removes a row from DataGridView dgv, where the idParam is the parameter string, and the idCell is the id row index of the dataGridView*
239. **private** **static** **void** removeRow(DataGridView dgv, **String** idParam, **int** idCell, SqlDataAdapter adapter)
240. {
241. Boolean notifyDelete = **false**; *//If data was deleted this flag will turn true*
242. **if** (dgv.SelectedRows.Count > 0) *//More than 1 selected row*
243. {
244. **int** toDelete = dgv.SelectedRows.Count; *//The amount to delete*
245. **while** (toDelete-- > 0) *//Whilst there is more to delete*
246. {
247. **int** indexToDelete = dgv.SelectedRows[toDelete].Index; *//Find index to delete*
248. **if** (FarmingDataSet.deleteFromTable(adapter, idParam, (**int**)dgv.Rows[indexToDelete].Cells[idCell].**Value**) == 1) *//Delete from database*
249. {
250. dgv.Rows.**Remove**(dgv.Rows[indexToDelete]); *//Remove from GUI*
251. notifyDelete = **true**; *//notify user*
252. }
253. }
254. **if** (notifyDelete) *//Notify user*
255. {
256. MessageBox.Show("The data was deleted sucessfully.");
257. }
258. }
259. **else** *//No selected rows*
260. {
261. MessageBox.Show("You have nothing selected to delete");
262. }
263. }
265. *//Checks for empty text boxes*
266. **public** Boolean handleNullOrWhitespace(TextBox box)
267. {
268. **if** (**String**.IsNullOrWhiteSpace(box.Text)) *//If the text box is empty notify user*
269. {
270. MessageBox.Show(box.Name + " cannot be empty.");
271. **return** **true**; *//Return true if empty*
272. }
273. **return** **false**; *//Return false if not empty*
274. }
275. }
276. }

# Github Changelog:

<https://github.com/crpaulino/FarmProgram/commits?author=sp1062>