Access Control Optimi College

Version: 1.0.0
Last Updated: [16 December 2023]
Maintainer: [Elizma le Grange]

Overview

The software programme called Access Control Optimi College was created in Microsoft Visual Studios with C#. The project made use of JavaScript, jQuery, SQL Database, and ASP.NET webforms to provide an effective and engaging user experience. The software's main goal is to make it easier to handle staff data in accordance with Optimi College's access control procedures. This involves running stored procedures, supplying customisable data outputs, and importing Excel files into the SQL Database.

Purpose

Import of Excel files to SQL Database:

The Users can import excel spreadsheets to SQL database from the import page.

Modified Data Export:

Enables users the capability to choose from a range of choices for creating reports that are customised to their needs.

Features

Architecture Details

- **Programming Language:** C#

Various methods in C# was used to create functionality of the user interface elements such as the dropdown list. To see full code please visit Github.

Please find below the code and explanation (the comments in code) for the drop downlist:

The main method is DropDownList_SelectOption_SelectedIndexChanged. Two methods were created for this method. Namely SetGridViewColumnsBasedOnSelectedOption and the BindGridViewBasedOnOption method.

DropDownList_SelectOption_SelectedIndexChanged:

```
1. //The DropDownList_SelectOption_SelectedIndexChanged method.
2.         protected void DropDownList_SelectOption_SelectedIndexChanged(object sender,
EventArgs e)
3.         {
4.             //Calls the SetGridViewColumnsBasedOnSelectedOption method.
5.             SetGridViewColumnsBasedOnSelectedOption();
6.
7.             //The data gets binds to the gridview based on the specific options or
conditions.
8.             BindGridViewBasedOnOption();
9.
10.            //The labelError will not be visible.
11.            LabelError.Visible = false;
12.        }
13.
```

SetGridViewColumnsBasedOnSelectedOption method:

```
1. //The SetGridViewColumnsBasedOnSelectedOption method.
2.         private void SetGridViewColumnsBasedOnSelectedOption()
3.         {
4.             // The value of the selected item is retrieved from the DropDownList named
selectedOption.
5.             string selectedOption = DropDownList_SelectOption.SelectedValue;
6.
7.             // The columns are all cleared, in the Gridview control named
Gridview_Employees.
8.             GridView_Employees.Columns.Clear();
9.
10.            //Below is a switch case to operate different cases based on which item is
selected in the dropdownlist.
11.            switch (selectedOption)
12.            {
13.                // The 'case' when the user selects the All Employees option from the
dropdown list.
14.                case "All Employees":
15.
16.                    // A new BoundField is created which is a class that represents the
columns in the gridview for the data field PersonnelID.
17.                    BoundField personnelIdColumn1 = new BoundField();   //The datasource
field.
18.                    personnelIdColumn1.DataField = "PersonnelID";       // Indicates the
specific field or attribute from the data source.
19.                    personnelIdColumn1.HeaderText = "Personnel ID";     // Defines the
text to be displayed as the title of the column.
20.                    GridView_Employees.Columns.Add(personnelIdColumn1); // Includes the
defined column within the GridView.
21.
22.                    BoundField cardNumberColumn1 = new BoundField();
23.                    cardNumberColumn1.DataField = "CardNumber";
24.                    cardNumberColumn1.HeaderText = "Card Number";
25.                    GridView_Employees.Columns.Add(cardNumberColumn1);
26.                    // The caseblock ends.
27.                    break;
28.
29.
30.                case "Fire Drill":
31.
32.                    BoundField dateAndTimeColumn2 = new BoundField();
33.                    dateAndTimeColumn2.DataField = "DateAndTime";
34.                    dateAndTimeColumn2.HeaderText = "Date And Time";
```

```
35.                     GridView_Employees.Columns.Add(dateAndTimeColumn2);
36.
37.                     BoundField personnelIdColumn2 = new BoundField();
38.                     personnelIdColumn2.DataField = "PersonnelID";
39.                     personnelIdColumn2.HeaderText = "Personnel ID";
40.                     GridView_Employees.Columns.Add(personnelIdColumn2);
41.
42.                     BoundField cardNumberColumn2 = new BoundField();
43.                     cardNumberColumn2.DataField = "CardNumber";
44.                     cardNumberColumn2.HeaderText = "Card Number";
45.                     GridView_Employees.Columns.Add(cardNumberColumn2);
46.
47.                     BoundField DeviceNameColumn2 = new BoundField();
48.                     DeviceNameColumn2.DataField = "DeviceName";
49.                     DeviceNameColumn2.HeaderText = "Device Name";
50.                     GridView_Employees.Columns.Add(DeviceNameColumn2);
51.
52.                     BoundField eventPointColumn2 = new BoundField();
53.                     eventPointColumn2.DataField = "EventPoint";
54.                     eventPointColumn2.HeaderText = "Event Point";
55.                     GridView_Employees.Columns.Add(eventPointColumn2);
56.
57.                     BoundField eventDescriptionColumn2 = new BoundField();
58.                     eventDescriptionColumn2.DataField = "EventDescription";
59.                     eventDescriptionColumn2.HeaderText = "Event Description";
60.                     GridView_Employees.Columns.Add(eventDescriptionColumn2);
61.                     break;
62.
63.
64.                 case "In & Out":
65.
66.                     BoundField dateAndTimeColumn3 = new BoundField();
67.                     dateAndTimeColumn3.DataField = "DateAndTime";
68.                     dateAndTimeColumn3.HeaderText = "Date And Time";
69.                     GridView_Employees.Columns.Add(dateAndTimeColumn3);
70.
71.                     BoundField personnelIdColumn3 = new BoundField();
72.                     personnelIdColumn3.DataField = "PersonnelID";
73.                     personnelIdColumn3.HeaderText = "Personnel ID";
74.                     GridView_Employees.Columns.Add(personnelIdColumn3);
75.
76.                     BoundField cardNumberColumn3 = new BoundField();
77.                     cardNumberColumn3.DataField = "CardNumber";
78.                     cardNumberColumn3.HeaderText = "Card Number";
79.                     GridView_Employees.Columns.Add(cardNumberColumn3);
80.
81.                     BoundField eventPointColumn3 = new BoundField();
82.                     eventPointColumn3.DataField = "EventPoint";
83.                     eventPointColumn3.HeaderText = "Event Point";
84.                     GridView_Employees.Columns.Add(eventPointColumn3);
85.
86.                     BoundField verifyTypeColumn3 = new BoundField();
87.                     verifyTypeColumn3.DataField = "VerifyType";
88.                     verifyTypeColumn3.HeaderText = "Verify Type";
89.                     GridView_Employees.Columns.Add(verifyTypeColumn3);
90.
91.
92.                     BoundField In_Out_StatusColumn3 = new BoundField();
93.                     In_Out_StatusColumn3.DataField = "In_OutStatus";
94.                     In_Out_StatusColumn3.HeaderText = "In/Out Status";
95.                     GridView_Employees.Columns.Add(In_Out_StatusColumn3);
96.
97.
98.                     BoundField eventDescriptionColumn3 = new BoundField();
99.                     eventDescriptionColumn3.DataField = "EventDescription";
100.                    eventDescriptionColumn3.HeaderText = "Event Description";
101.                    GridView_Employees.Columns.Add(eventDescriptionColumn3);
102.                    break;
103.
104.                case "Late Today":
```

```
105.                     BoundField dateAndTimeColumn4 = new BoundField();
106.                     dateAndTimeColumn4.DataField = "DateandTime";
107.                     dateAndTimeColumn4.HeaderText = "Date And Time";
108.                     GridView_Employees.Columns.Add(dateAndTimeColumn4);
109.
110.                     BoundField personnelIdColumn4 = new BoundField();
111.                     personnelIdColumn4.DataField = "PersonnelID";
112.                     personnelIdColumn4.HeaderText = "Personnel ID";
113.                     GridView_Employees.Columns.Add(personnelIdColumn4);
114.
115.                     BoundField FirstLogInTimeColumn4 = new BoundField();
116.                     FirstLogInTimeColumn4.DataField = "FirstLogInTime";
117.                     FirstLogInTimeColumn4.HeaderText = "First LogIn Time";
118.                     GridView_Employees.Columns.Add(FirstLogInTimeColumn4);
119.
120.                     BoundField lateTodayColumn = new BoundField();
121.                     lateTodayColumn.DataField = "LateToday";
122.                     lateTodayColumn.HeaderText = "Late Today";
123.                     GridView_Employees.Columns.Add(lateTodayColumn);
124.                     break;
125.
126.
127.                 case "Sign In":
128.
129.                     BoundField dateAndTimeColumn5 = new BoundField();
130.                     dateAndTimeColumn5.DataField = "DateAndTime";
131.                     dateAndTimeColumn5.HeaderText = "Date And Time";
132.                     GridView_Employees.Columns.Add(dateAndTimeColumn5);
133.
134.                     BoundField personnelIdColumn5 = new BoundField();
135.                     personnelIdColumn5.DataField = "PersonnelID";
136.                     personnelIdColumn5.HeaderText = "Personnel ID";
137.                     GridView_Employees.Columns.Add(personnelIdColumn5);
138.
139.                     BoundField cardNumberColumn5 = new BoundField();
140.                     cardNumberColumn5.DataField = "CardNumber";
141.                     cardNumberColumn5.HeaderText = "Card Number";
142.                     GridView_Employees.Columns.Add(cardNumberColumn5);
143.
144.                     BoundField eventPointColumn5 = new BoundField();
145.                     eventPointColumn5.DataField = "EventPoint";
146.                     eventPointColumn5.HeaderText = "Event Point";
147.                     GridView_Employees.Columns.Add(eventPointColumn5);
148.
149.                     BoundField eventDescriptionColumn5 = new BoundField();
150.                     eventDescriptionColumn5.DataField = "EventDescription";
151.                     eventDescriptionColumn5.HeaderText = "Event Description";
152.                     GridView_Employees.Columns.Add(eventDescriptionColumn5);
153.                     break;
154.
155.                 case "Sign Out":
156.
157.                     BoundField dateAndTimeColumn6 = new BoundField();
158.                     dateAndTimeColumn6.DataField = "DateAndTime";
159.                     dateAndTimeColumn6.HeaderText = "Date And Time";
160.                     GridView_Employees.Columns.Add(dateAndTimeColumn6);
161.
162.                     BoundField personnelIdColumn6 = new BoundField();
163.                     personnelIdColumn6.DataField = "PersonnelID";
164.                     personnelIdColumn6.HeaderText = "Personnel ID";
165.                     GridView_Employees.Columns.Add(personnelIdColumn6);
166.
167.                     BoundField DeviceNameColumn6 = new BoundField();
168.                     DeviceNameColumn6.DataField = "DeviceName";
169.                     DeviceNameColumn6.HeaderText = "DeviceName";
170.                     GridView_Employees.Columns.Add(DeviceNameColumn6);
171.
172.                     BoundField eventPointColumn6 = new BoundField();
173.                     eventPointColumn6.DataField = "EventPoint";
174.                     eventPointColumn6.HeaderText = "Event Point";
```

```
175.                     GridView_Employees.Columns.Add(eventPointColumn6);
176.
177.                     BoundField verifyTypeColumn6 = new BoundField();
178.                     verifyTypeColumn6.DataField = "VerifyType";
179.                     verifyTypeColumn6.HeaderText = "Verify Type";
180.                     GridView_Employees.Columns.Add(verifyTypeColumn6);
181.
182.                     BoundField eventDescriptionColumn6 = new BoundField();
183.                     eventDescriptionColumn6.DataField = "EventDescription";
184.                     eventDescriptionColumn6.HeaderText = "Event Description";
185.                     GridView_Employees.Columns.Add(eventDescriptionColumn6);
186.                     break;
187.
188.
189.                 case "Error Records":
190.
191.                     BoundField dateAndTimeColumn7 = new BoundField();
192.                     dateAndTimeColumn7.DataField = "DateAndTime";
193.                     dateAndTimeColumn7.HeaderText = "Date And Time";
194.                     GridView_Employees.Columns.Add(dateAndTimeColumn7);
195.
196.                     BoundField DeviceNameColumn7 = new BoundField();
197.                     DeviceNameColumn7.DataField = "DeviceName";
198.                     DeviceNameColumn7.HeaderText = "DeviceName";
199.                     GridView_Employees.Columns.Add(DeviceNameColumn7);
200.
201.                     BoundField eventPointColumn7 = new BoundField();
202.                     eventPointColumn7.DataField = "EventPoint";
203.                     eventPointColumn7.HeaderText = "Event Point";
204.                     GridView_Employees.Columns.Add(eventPointColumn7);
205.
206.                     BoundField verifyTypeColumn7 = new BoundField();
207.                     verifyTypeColumn7.DataField = "VerifyType";
208.                     verifyTypeColumn7.HeaderText = "Verify Type";
209.                     GridView_Employees.Columns.Add(verifyTypeColumn7);
210.
211.                     BoundField In_Out_StatusColumn7 = new BoundField();
212.                     In_Out_StatusColumn7.DataField = "InOutStatus";
213.                     In_Out_StatusColumn7.HeaderText = "In/Out Status";
214.                     GridView_Employees.Columns.Add(In_Out_StatusColumn7);
215.
216.                     BoundField eventDescriptionColumn7 = new BoundField();
217.                     eventDescriptionColumn7.DataField = "EventDescription";
218.                     eventDescriptionColumn7.HeaderText = "Event Description";
219.                     GridView_Employees.Columns.Add(eventDescriptionColumn7);
220.                     break;
221.
222.                 case "All Records":
223.
224.                     BoundField dateAndTimeColumn8 = new BoundField();
225.                     dateAndTimeColumn8.DataField = "DateAndTime";
226.                     dateAndTimeColumn8.HeaderText = "Date And Time";
227.                     GridView_Employees.Columns.Add(dateAndTimeColumn8);
228.
229.                     BoundField personnelIdColumn8 = new BoundField();
230.                     personnelIdColumn8.DataField = "PersonnelID";
231.                     personnelIdColumn8.HeaderText = "Personnel ID";
232.                     GridView_Employees.Columns.Add(personnelIdColumn8);
233.
234.                     BoundField cardNumberColumn8 = new BoundField();
235.                     cardNumberColumn8.DataField = "CardNumber";
236.                     cardNumberColumn8.HeaderText = "Card Number";
237.                     GridView_Employees.Columns.Add(cardNumberColumn8);
238.
239.                     BoundField DeviceNameColumn8 = new BoundField();
240.                     DeviceNameColumn8.DataField = "DeviceName";
241.                     DeviceNameColumn8.HeaderText = "Device Name";
242.                     GridView_Employees.Columns.Add(DeviceNameColumn8);
243.
244.                     BoundField eventPointColumn8 = new BoundField();
```

```
                    eventPointColumn8.DataField = "EventPoint";
                    eventPointColumn8.HeaderText = "Event Point";
                    GridView_Employees.Columns.Add(eventPointColumn8);

                    BoundField verifyTypeColumn8 = new BoundField();
                    verifyTypeColumn8.DataField = "VerifyType";
                    verifyTypeColumn8.HeaderText = "Verify Type";
                    GridView_Employees.Columns.Add(verifyTypeColumn8);

                    BoundField In_Out_StatusColumn8 = new BoundField();
                    In_Out_StatusColumn8.DataField = "InOutStatus";
                    In_Out_StatusColumn8.HeaderText = "In/Out Status";
                    GridView_Employees.Columns.Add(In_Out_StatusColumn8);

                    BoundField eventDescriptionColumn8 = new BoundField();
                    eventDescriptionColumn8.DataField = "EventDescription";
                    eventDescriptionColumn8.HeaderText = "Event Description";
                    GridView_Employees.Columns.Add(eventDescriptionColumn8);

                    BoundField RemarksColumn8 = new BoundField();
                    RemarksColumn8.DataField = "Remarks";
                    RemarksColumn8.HeaderText = "Remarks";
                    GridView_Employees.Columns.Add(RemarksColumn8);
                    break;


                default:
                    break;
            }

public void TestMethod()
{


    Console.WriteLine("Select this line and click Highlight line button");
}

```

```
 1. //This method will bind data to the gridview based on the selected options in the
dropdownlist.
 2.         private void BindGridViewBasedOnOption()
 3.         {
 4.             //The selected value is retrieved from the Dropdownlist.
 5.             string selectedOption = DropDownList_SelectOption.SelectedValue;
 6.             //The connection string is established to connect to the SQL database.
 7.             string connectionString =
ConfigurationManager.ConnectionStrings["Optimi_CollegeConnectionString"].ConnectionString;
 8.
 9.             //Below is a switch case code to switch between the selected options in the
dropdownlist.
10.             try
11.             {
12.                 //Database connection to SQL is established.
13.                 using (SqlConnection connection = new SqlConnection(connectionString))
14.                 {
15.                     //The connection is opened.
16.                     connection.Open();
17.                     // Set up a blank string variable for queries and create a list to
store dynamic objects.
18.                     string query = string.Empty;
19.                     var employees = new List<dynamic>();
20.
21.
22.                     //The query to be executed is determined based on the selectedOption.
23.                     switch (selectedOption)
24.                     {
```

```
                        //The queries are specified to correspond to various options.
                        //Below all the stored procedures are called from SQL Database.
                        case "All Employees":
                            query = "EXEC AllEmployees";
                            break;
                        case "Fire Drill":
                            query = "EXEC FireDrill";
                            break;
                        case "In & Out":
                            query = "EXEC EmployeeInAndOut";
                            break;
                        case "Late Today":
                            query = "EXEC LateEmployees";
                            break;
                        case "Sign In":
                            query = "EXEC EmployeeSignIn";
                            break;
                        case "Sign Out":
                            query = "EXEC EmployeeSignOut";
                            break;
                        case "Error Records":
                            query = "EXEC ErrorRecords";
                            break;
                        case "All Records":
                            query = "EXEC ShowAllRecords";
                            break;

                        default:
                            break;
                    }

                    // The query needs to run if it contains valid content.
                    if (!string.IsNullOrEmpty(query))
                    {
                        //Run the query and get information into the 'employees' list.
                        employees = connection.Query(query).ToList();
                    }
                    //Use the collected information to fill up the GridView and display
it.
                    GridView_Employees.DataSource = employees;
                    GridView_Employees.DataBind();
                }
            }
            catch (Exception ex)
            {
                //If something goes wrong, show the error message on the LabelError area.
                // ""Error: " + ex.Message;" If any error occurs the error, followed by
the message will display to the user.
                LabelError.Visible = true;
                LabelError.Text = "Error: " + ex.Message;
            }
        }

```

- **Development Environment:** Microsoft Visual Studios
- **Web Framework:** ASP.NET Webforms
- **Database Management System:** SQL Database
- **Database Schema:** Optimi_College (Includes a table named ImportEmployee)

- **Stored Procedures:** AllEmployees, DeleteDuplicateRecords, EmployeeInAndOut, EmployeeSignIn, EmployeeSignOut, ErrorRecords, FireDrill, ImportAndSortEmployeeRecords, LateEmployees, ShowAllRecords.

Late Today Stored Procedure:

```
USE [Optimi_College]
GO
/****** Object:  StoredProcedure [dbo].[LateEmployees]    Script Date: 18 Dec 2023
04:57:24 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:        <Author: Elizma le Grange>
-- Create date:   <Create Date: 2 Desember 2023>
-- Description:   <Description: This is a Stored Procedure to show all the late
employees signing in the first time after 8:30 am>
-- =============================================
ALTER PROCEDURE [dbo].[LateEmployees]
                                                -- The Stored Procedure name,
LateEmployees.
AS
BEGIN
    SET NOCOUNT ON;

                                                                            -- SET
NOCOUNT ON added to prevent extra result sets from
--

interfering with SELECT statements.

    DECLARE @thresholdTime TIME = '08:30:00.000';
                                           -- The time at which the employee should
start are declared.

    SELECT
        CONVERT(DATETIME, [Date And Time]) AS [DateAndTime],
                              -- The Date And Time column is selected and given an
alias of DateAndTime, which will be used in C# to call this procedure.
        [Personnel_ID] AS PersonnelID,
                                                -- The Personnel_ID column is
selected and given an alias of PersonnelID, which will be used in C# to call this
procedure.
        [Date And Time] AS FirstLogInTime,
                                                -- The Date And Time column is
selected and given an alias of FirstLogInTime, which will be used in C# to call this
procedure.
        'Late' AS LateToday
                                                        -- The Value 'Late' is
created with alias Late too show the late employees.

    FROM (
        SELECT
            [Personnel_ID],
                                                        -- The personnel_ID column is
selected from the Optimi_College database from ImportEmployee table
            [Date And Time],
                                                        -- The [Date And Time] column
is selected from the Optimi_College database from ImportEmployee table
```

```
            ROW_NUMBER() OVER (PARTITION BY CONVERT(DATE, [Date And Time]),
[Personnel_ID]       -- Row numbers are generated starting from 1 for

                                                                        -- Each
unique combination of date and personnelID ordered by the date and time

                --
                     ORDER BY [Date And Time]) AS RowNum
                                                          -- The Data and Time and
Personnel_ID wil be ordered by Data and Time.

        FROM
            [Optimi_College].[dbo].[ImportEmployee]
                                          -- The records are selected from
Otimi_College database and the table, Import Employee


        WHERE [Event_Description] NOT IN (
                                                -- If the Event_Description column
has the values:Person Not Registered, Validate Interval TooShort

                                                                -- Disconnected and
exit Button Open, it will be excluded from the results.
                'Person Not Registered',
                'Validate Interval Too Short',
                'Disconnected',
                'Exit Button Open'
            )
            AND CONVERT(TIME, [Date And Time]) > @thresholdTime
                                    -- The records are filtered where Login time is
greater than @thresholdTime
    ) AS Earliest_LogIn_Employee
                                                -- My subquery is
Earliest_LogIn_Employee which identifies the earliest late logins.

    WHERE  Earliest_LogIn_Employee.RowNum = 1
                                          -- This will filter the late login records
for each employee on a specific day

    ORDER BY [Date And Time] ASC;
                                                -- Sorting the records in ascending
order by Date and Time.


END
```

Functionalities

Import Page

- **File Import:** Users can select the Excel files via a user-friendly interface, an upload box.
- **Data Display:** Displays the uploaded Excel data in a Gridview to view before the user imports to the database.
- **Import to Database:** Allows users to import the excel file data into the SQL Database into the ImportEmployee table.

Export Page

- **Options Selection:** Users can choose from a range of options to generate customized reports.
- **Date and Personnel ID Filtering:** Allows users to filter records based on selected start and end dates as well as personnel ID for precise data retrieval.
- **Excel Export:** Facilitates the download of generated reports in Excel format for further analysis.

User Interface Flow

Import Page

The initial landing page for users to upload Excel files and import data is called the Import page.

Export Page

Subsequent page allowing users to select export options, filter records, and generate custom reports.

Installation Requirements

SQL Server 2019

Windows 10 TH1 1507 or greater

6 GB of available hard-disk space. (Disk space may vary depending on which components the user may install.

Recommended memory: 4GB

Processor speed: min (x64 1.4 GHz)

Recommended (2.0 GHz or faster)

Processor type: x64 Processor, AMD Opteron, AMD Athlon 64, Intel Xeon with Intel EM64T support, Intel Pentium IV with EM64T support

Installation Steps

Please follow the recommended steps of SQL Server 2019 software as pre scribed by software company.

Usage

Please refer to the Help.pdf

Importing Employee Records

1. Navigate to the Import Page.
2. Choose an Excel file to upload.
3. Review the uploaded data in the Gridview.
4. Click the Import button to import the data into the SQL Database.

Exporting Customized Reports

1. Go to the Export Page.
2. Select an option from the dropdown list.
3. Apply date and personnel ID filters if needed.
4. Click the Export button to download the generated report in Excel format.

Troubleshooting

Please refer to TroubleShooting.pdf

Common Issues

- **No File Selected for Import:**
    - **Issue:** Users encounter an error message stating, "Please select an Excel file to upload!!!"
- **No Data to Upload:**
    - **Issue:** Users encounter an error message indicating, "No data to upload to SQL Server."

Troubleshooting Steps

If users encounter any of these errors, follow these steps:

1. **No File Selected for Import:**
    - **Resolution:** Refresh the page to remove the error message. Ensure an Excel file is selected before clicking the load button.
2. **No Data to Upload:**

- **Resolution:** Refresh the page to remove the error message. Ensure a valid file with data is selected for import.