Spring 2016

GSU Schedule File Transformation Tools

Pratima Dharmala
Governors State University

Radhika Eedara
Governors State University

Phanendar Movva
Governors State University

ABSTRACT

GSU class schedule can be printed or saved as a text file. This file is not very user friendly to analyze the data and take effective decisions. This text file has section details spanned across multiple pages with headers repeating in each page. This text file also has some additional details in between sections that are not necessarily used. This project "***GSU Schedule File Transformation Tools***" focuses on creating an online application that provides workable spreadsheet from GSU section schedule text file. This will be done by extracting data from text file, transform the data and load it into excel. Transformed data will also be stored in database for future reference, in case of excel or text files are lost. This project works like online ETL tool for GSU members and provides business intelligent data. Transformed data will help professors, student advisors and students in many ways by simply using the filters in a single glance. Some of the benefits include

1. Current availability across multiple sections
2. To see which sections are filling up fast
3. Waiting list details of different sections
4. Planning sections for any student by days of the week.

Table of Content

List of Figures

1 Project Description

GSU class schedule allows every student to plan their semester and register for the subjects. All the available courses are displayed under the registration section in the GSU website. The registration section is categorized into various sections, where the available seats for registering the course or section to drop the registered subjects can be found. In this case a student or the professor can also save the available term schedule either in a text file or print it as a pdf document. These files are saved under this criteria can help either professor or the student to find the available courses. Considering the text files that are saved by one of the clients, a simple and unique application is designed, where-in-which the text file is converted into excel file with the updated fields. This tool is very handy for the academic advisor to update the sections consisting of many columns. The columns contain Course subject, Faculty name, Start date, End date, Number of seats available, Number of students registered and Term availability.

The motto of this application is to convert the saved text file into an excel file that contains all the sections as mentioned above. Using the principle of ETL, the text file is transformed into excel file and uploaded into the database or can be downloaded onto a local machine. Only registered users or users that are stored into the database are able to login into this application. The application is designed using conversion method for file transformation. This application is user friendly with GUI for the user or administrator to access the files and database. When the user logins into the application the home page consists multiple options, by entering the term in the text box provided a user can load the text file saved in the database or the local machine. When the text is loaded it displays in the center of page and the text is ready for conversion. With the available Transform to excel button the file is transformed into excel successfully and then is ready for download that is stored in the database.

1.1 Assumptions and Dependencies

- We assume that input text file format will not change to ensure the application work with no issues.
- Text file needs to be placed in the root of the project to be picked automatically.
- Online course information should have Days of Week, Start and End dates as NA.
- If faculty is not determined for the course, it should be passed as TBD or Not Available.
- Any changes to the text file format like below need thorough impact analysis
    - adding new fields to the text
    - change in location of fields
    - change in the field separator on text file

1.2 Future Enhancements

Input text file is currently manually located in the root directory of the project. This application may very well be enhanced in the future to pick the file from any location of the computer. As the data grows, database may need to be normalized and indexes need to be created for faster access of data.

1.3 Definitions and Abbreviations

- ETL: Extract Transform Load
- SQL: Structured Query language
- DTS: Data Transformation Services
- GUI: Graphical User Interface
- ADO: ActiveX Data Objects
- OLED: Object Linking and Embedding, Database
- ORM: Object-Relational Mapping
- CPU: Central Processing Unit
- DB: Database
- OLAP: Online Analytical Processing
- ODI: Oracle Data Integrator
- XML: Extensible Markup Language
- IIS: Internet Information Services

2 *Project Technical Description*

Description:
Data Transformation Services, or DTS, is generally a group of objects and services that allows the computerization of extraction, transformation and load actions to or from a database. These are DTS objects or packages and their components, and the utilities are called DTS tools. DTS was incorporated with earlier versions of Microsoft SQL Server, and also, was quite often utilized with SQL Server databases, while it could be used autonomously with other databases.
In registering, Extract, Transform and Load (ETL) alludes to a procedure in database utilization and particularly in information warehousing that:
Separates information from homogeneous or heterogeneous information sources. Changes the information for putting away it in the best possible configuration or structure for the motivations behind questioning and examination. Loads it into the last target (database, all the more particularly, operational information store, information shop, or information distribution center)
Typically all the three stages execute in parallel subsequent to the information extraction requires some serious energy, so while the information is being pulled another change process executes, handling the officially got information and readies the information for stacking and when there is a few information prepared to be stacked into the objective, the information stacking commences without sitting tight for the fulfillment of the past stages.
ETL frameworks generally coordinate information from various applications (frameworks), regularly created and upheld by various sellers or facilitated on independent PC equipment. The different frameworks containing the first information are as often as possible oversaw and worked by various representatives. For instance, a cost bookkeeping framework may join information from finance, deals, and buying.

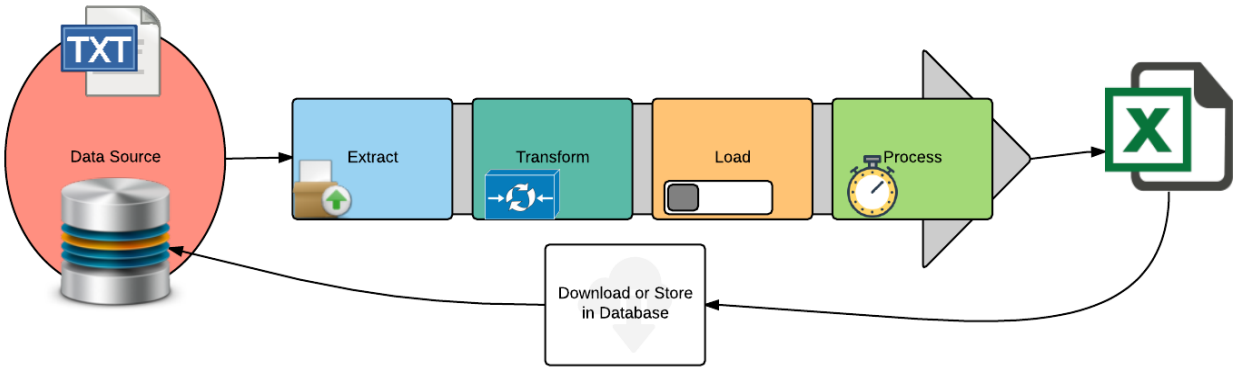


Fig. 1 Data flow

The primary objective of Extracting is to off-burden the information from the source frameworks as quick as could be expected under the circumstances and as less unwieldy for these source frameworks, its improvement group and its end-clients as could be expected under the circumstances. This suggests the sort of source framework and its qualities – OLTP framework, OLTP legacy information, numerous occasions, old Data Warehouse, files, altered and variable outer information, spreadsheets … - ought to be considered however much as could be expected. Besides it likewise infers that the most relevant extraction strategy ought to be picked – source date/time stamps, database triggers, database log tables, different delta components, full versus incremental invigorate, delta parameterization, controlled overwrite, mixture – relying upon the circumstance.
Change and Loading the information is about coordinating lastly moving the incorporated information to the presentation zone which can be gotten to through front-end devices by the end-client group. Here the accentuation ought to be on truly utilizing the offered usefulness by the picked ETL-instrument and utilizing it as a part of the best way. It is insufficient to just utilize an ETL-instrument, but rather still utilize different indirect accesses which don't amplify the utilization of the apparatus. Besides, in a medium to vast scale information stockroom environment it is vital to institutionalize (measurement and reality mappings) however much as could reasonably be expected as opposed to going for customization. This will decrease the throughput time of the distinctive source-to-target advancement exercises which shape the majority of the conventional ETL exertion. A reaction in a later stage is the era of the purported ETL-scripts taking into account this institutionalization and pre-characterized metadata. At last,

the distinctive individual mappings or employments ought to go for re-ease of use and understandability bringing about sufficiently little pieces which can be effortlessly repaired or tried so far as that is concerned.

2.1 ***Application Architecture***

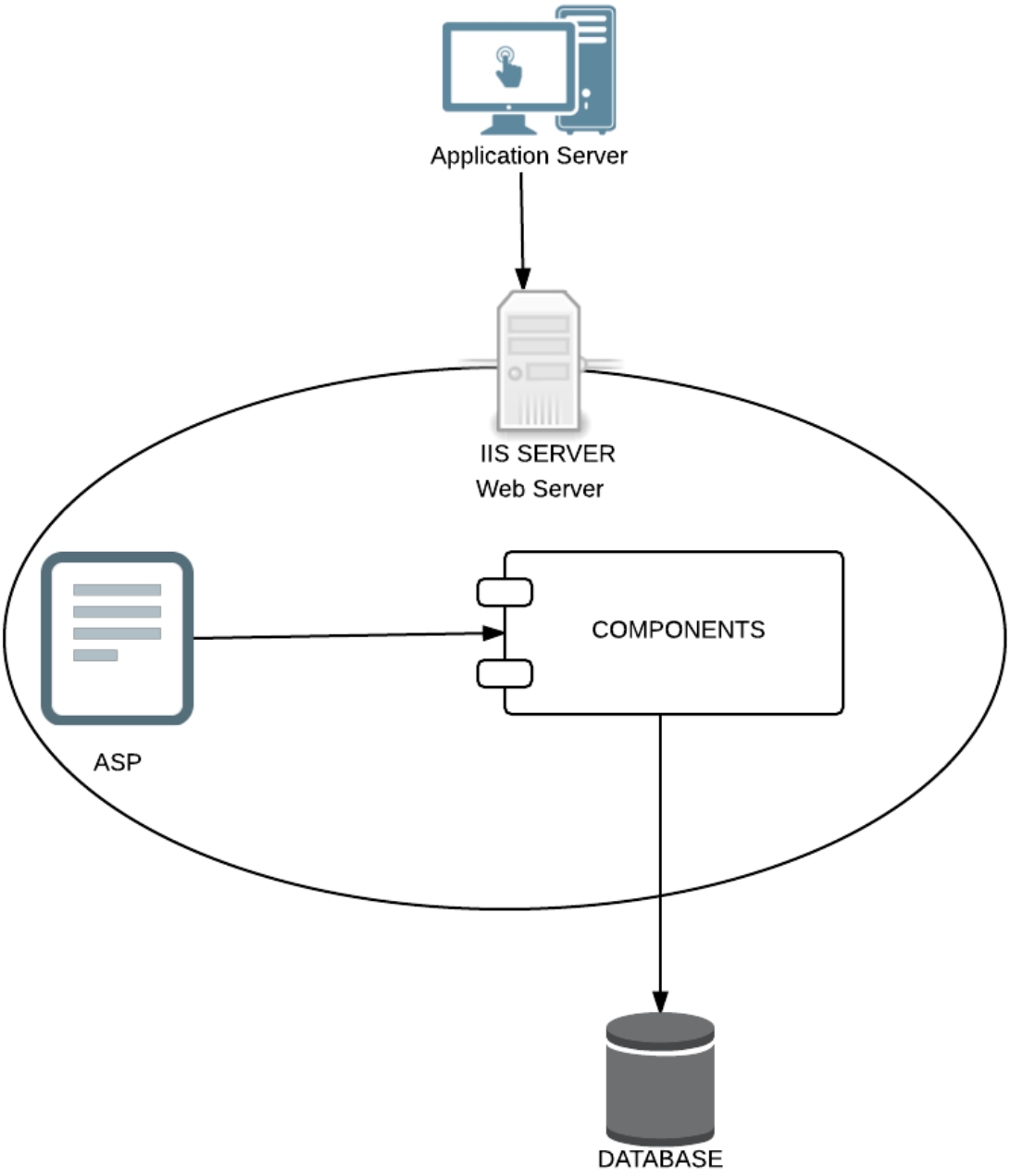


Fig. 2 Architecture

One-Time Extract. This architecture is designed for one time extract of the client text file into .xml (excel) file. The whole web application is designed in Visual studio 2012 using default libraries. The files used to design this application are mentioned below.

2.2 *Application Information flows*

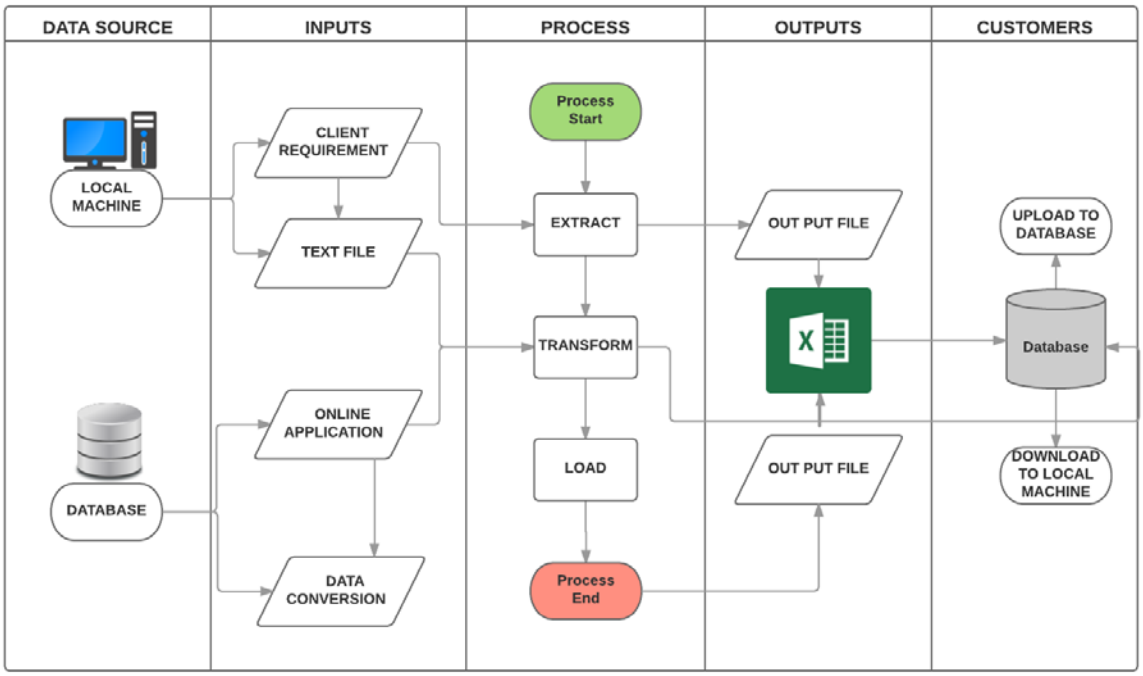


Fig. 3 Flow of data in Application

2.3 *Data Model*

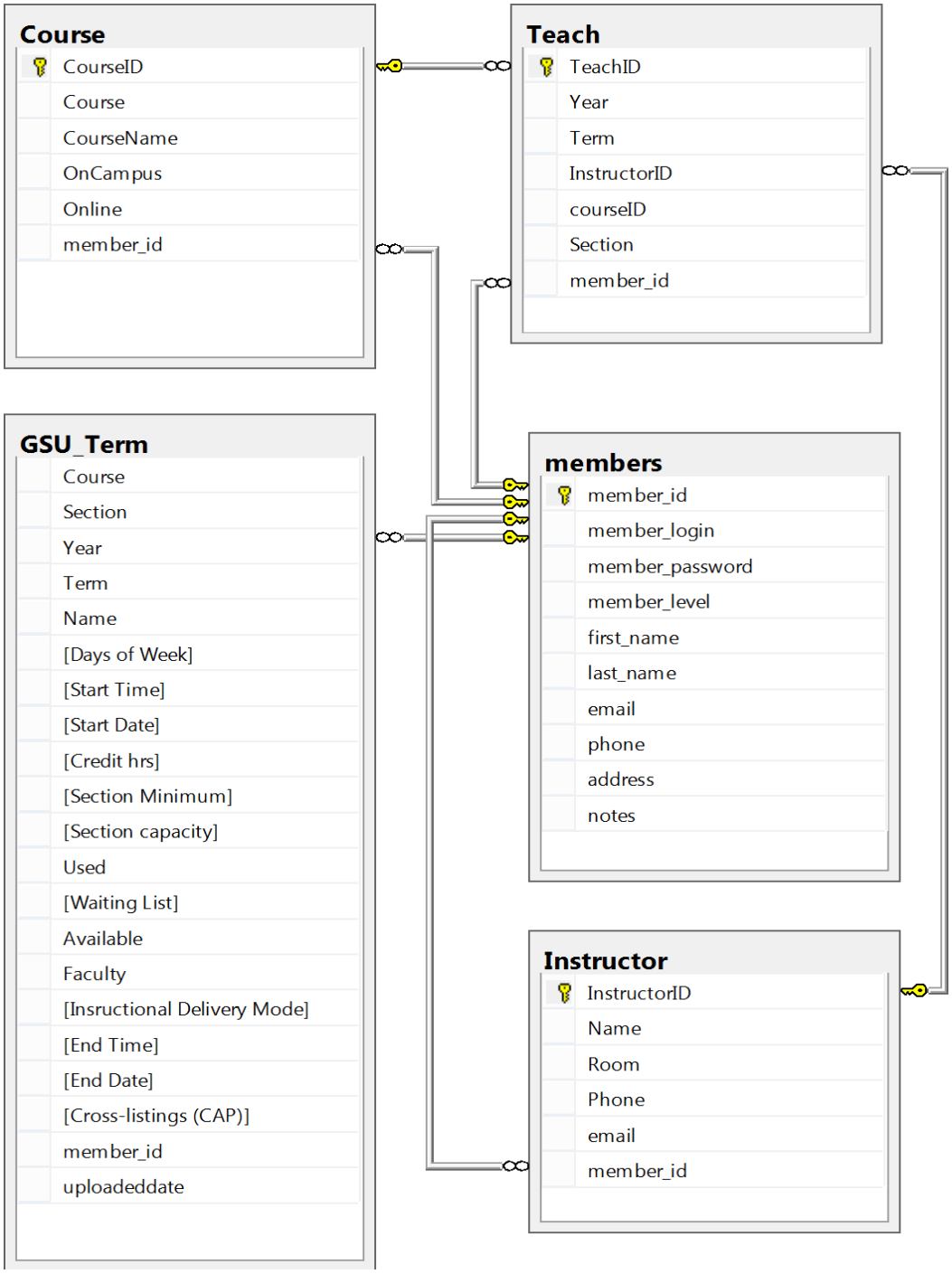


Fig. 4 Database Diagram

2.4 ***Screenshots***

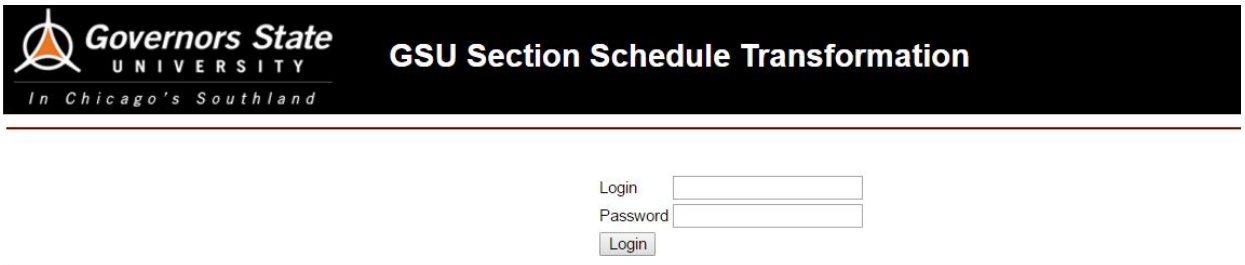


Fig. 5– Login Page

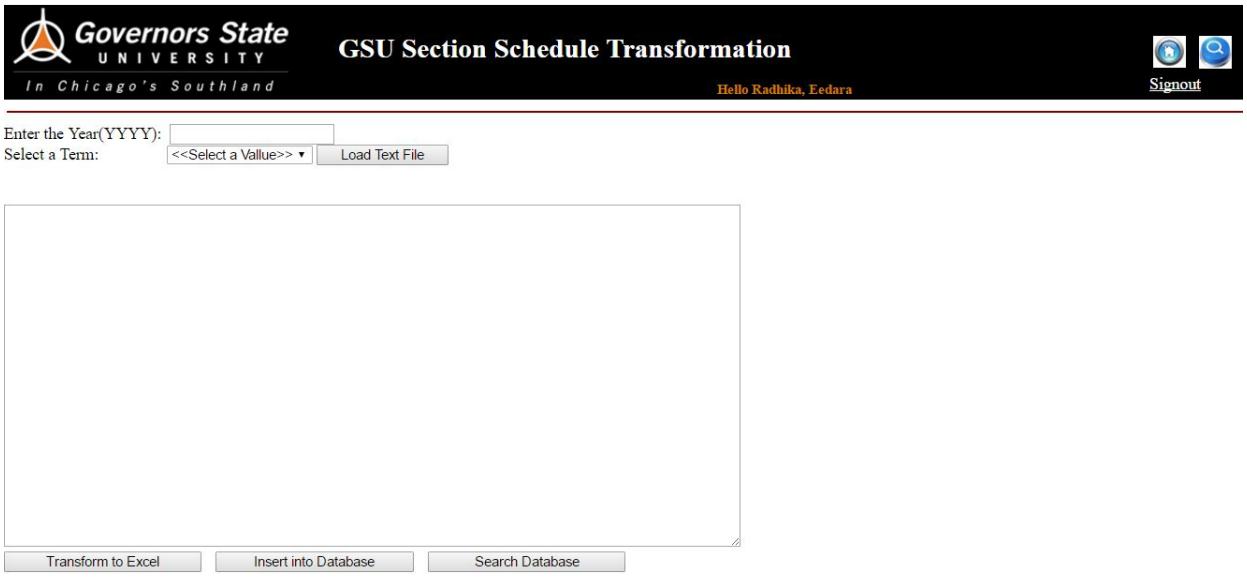


Fig. 6– Home Page

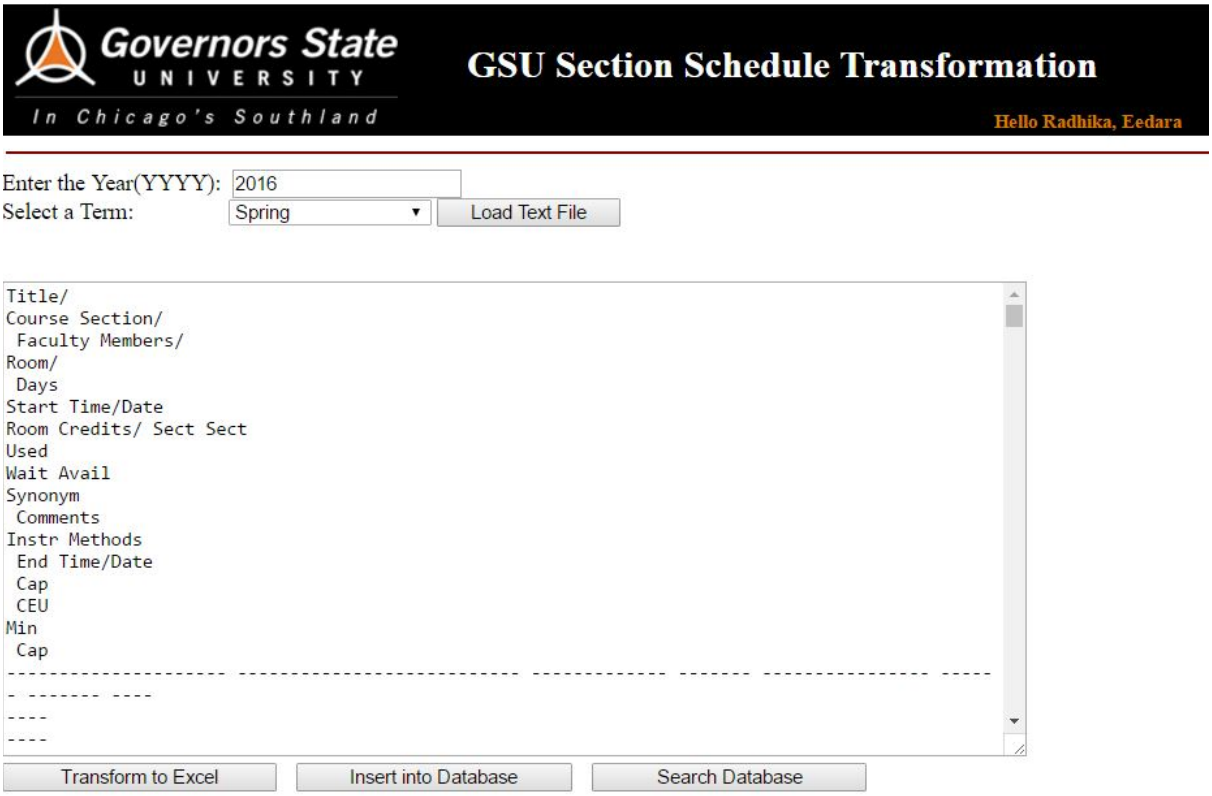


Fig. 7– Text Loading

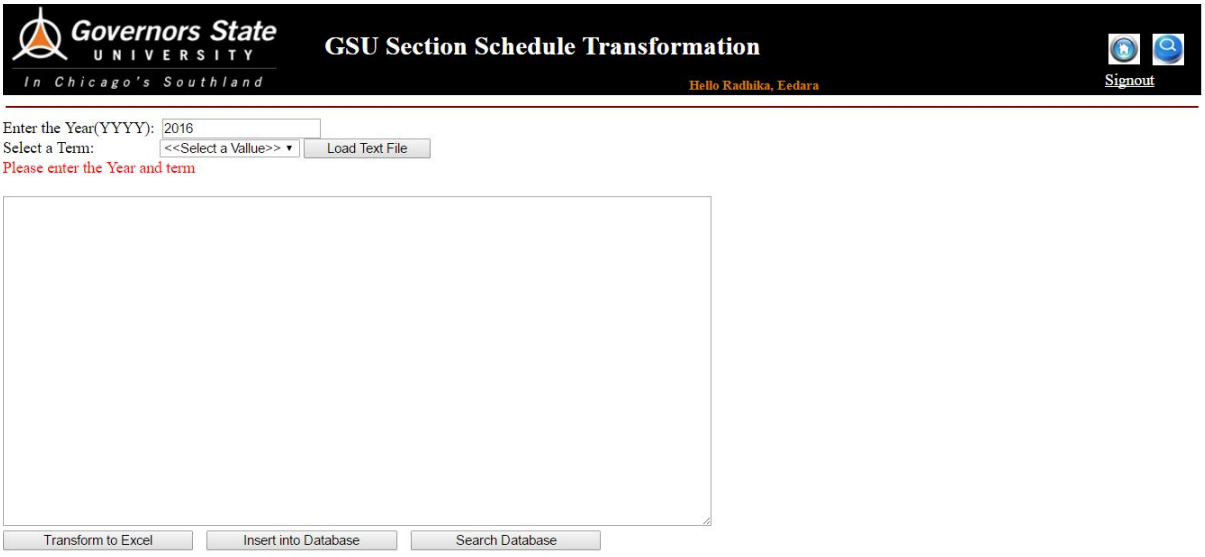
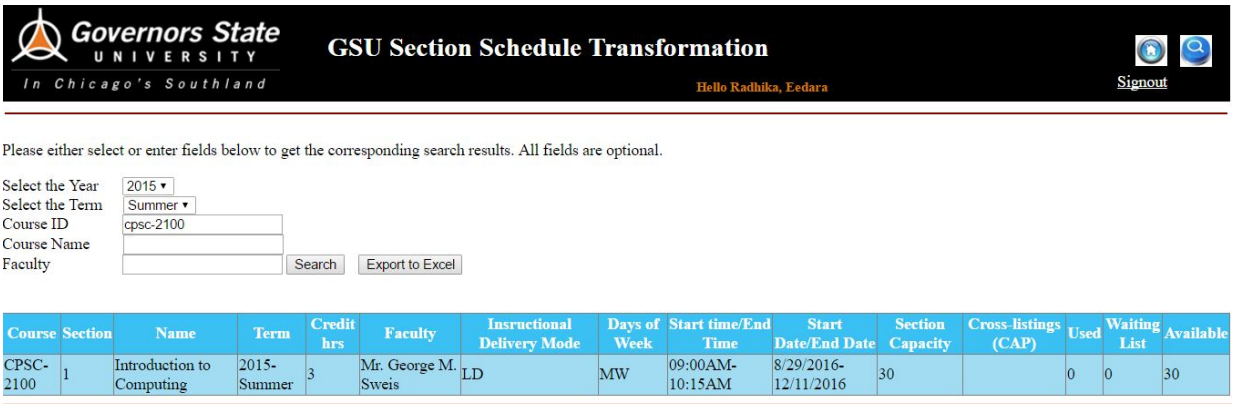


Fig. 8– Error Message



Course	Section	Name	Term	Credit hrs	Faculty	Insructional Delivery Mode	Days of Week	Start time/End Time	Start Date/End Date	Section Capacity	Cross-listings (CAP)	Used	Waiting List	Available
CPSC-2100	1	Introduction to Computing	2015-Summer	3	Mr. George M. Sweis	LD	MW	09:00AM-10:15AM	8/29/2016-12/11/2016	30		0	0	30

Fig. 9– Search by Course ID

Course	Section	Name	Term	Credit hrs	Faculty	Insructional Delivery Mode	Days of Week	Start time/End Time	Start Date/End Date	Section Capacity	Cross-listings (CAP)	Used	Waiting List	Available
CPSC-2005	1	Introduction to Computer	2015-Summer	3	Mr. George M. Sweis	LD	MW	11:00AM-12:15PM	8/29/2016-12/11/2016	30		0	0	30
CPSC-2005	2	Introduction to Computer	2015-Summer	3	Mr. Aslam Shahid	LD	MW	02:00PM-03:15PM	8/29/2016-12/11/2016	30		0	0	30
CPSC-3148	1	Computer Programming: Java	2015-Summer	3	Dr. Xueqing Tang	LD	MW	12:30PM-01:45PM	8/29/2016-12/11/2016	30		0	0	30
CPSC-3148	2	Computer Programming: Java	2015-Summer	3	Mr Do Young Park	LD	T	04:30PM-07:20PM	8/30/2016-12/11/2016	40	CPSC-6548(20)	0	0	40
CPSC-4205	1	Computer Organization	2015-Summer	3	Dr Nana Amponsah	LD	R	04:30PM-07:20PM	9/1/2016-12/11/2016	30		0	0	30
CPSC-4342	1	Intro to Computer Networks	2015-Summer	3	Mr Richard Manprisi	LD	MW	11:00AM-12:15PM	8/29/2016-12/11/2016	30		0	0	30

Fig. 10– Search by Course Name

Governors State UNIVERSITY
In Chicago's Southland

GSU Section Schedule Transformation

Hello Radhika, Eedara

Signout

Please either select or enter fields below to get the corresponding search results. All fields are optional.

Select the Year 2015
Select the Term Summer
Course ID
Course Name computer
Faculty park Search Export to Excel

Course	Section	Name	Term	Credit hrs	Faculty	Insructional Delivery Mode	Days of Week	Start time/End Time	Start Date/End Date	Section Capacity	Cross-listings (CAP)	Used	Waiting List	Available
CPSC-3148	2	Computer Programming: Java	2015-Summer	3	Mr Do Young Park	LD	T	04:30PM-07:20PM	8/30/2016-12/11/2016	40	CPSC-6548(20)	0	0	40
CPSC-6548	2	Computer Programming: Java	2015-Summer	3	Mr Do Young Park	LD	T	04:30PM-07:20PM	8/30/2016-12/11/2016	40	CPSC-6548(20)	0	0	40
CPSC-6548	3	Computer Programming: Java	2015-Summer	3	Mr Do Young Park	LD	W	04:30PM-07:20PM	8/31/2016-12/11/2016	40		0	0	40
CPSC-6548	4	Computer Programming: Java	2015-Summer	3	Mr Do Young Park	LD	T	12:30PM-03:20PM	8/30/2016-12/11/2016	40		0	0	40

Fig. 11– Search by Faculty Name

Governors State UNIVERSITY
In Chicago's Southland

GSU Section Schedule Transformation

Hello Radhika, Eedara

Signout

Please either select or enter fields below to get the corresponding search results. All fields are optional.

Select the Year 2015
Select the Term All
Course ID
Course Name
Faculty Search Export to Excel

Course	Section	Name	Term	Credit hrs	Faculty	Insructional Delivery Mode	Days of Week	Start time/End Time	Start Date/End Date	Section Capacity	Cross-listings (CAP)	Used	Waiting List	Available
IT-4520	1	IP Routing	2015-Summer	3	Mr Stephen Hyzny	LD	R	04:30PM-07:20PM	9/1/2016-12/11/2016	20		0	0	20
CPSC-2005	1	Introduction to Computer	2015-Summer	3	Mr. George M. Sweis	LD	MW	11:00AM-12:15PM	8/29/2016-12/11/2016	30		0	0	30
CPSC-2005	2	Introduction to Computer	2015-Summer	3	Mr. Aslam Shahid	LD	MW	02:00PM-03:15PM	8/29/2016-12/11/2016	30		0	0	30
CPSC-2100	1	Introduction to Computing	2015-Summer	3	Mr. George M. Sweis	LD	MW	09:00AM-10:15AM	8/29/2016-12/11/2016	30		0	0	30
CPSC-3099	1	Junior Seminar	2015-Summer	3	Mr Stephen Hyzny	LD	M	07:30PM-10:20PM	8/29/2016-12/11/2016	80	CPSC-3099(40)	0	0	80
CPSC-	1	Visual BASIC	2015-	3	Mr. George	LD	F	09:00AM-	9/2/2016-	30		0	0	30

Fig. 12 – Search by Year

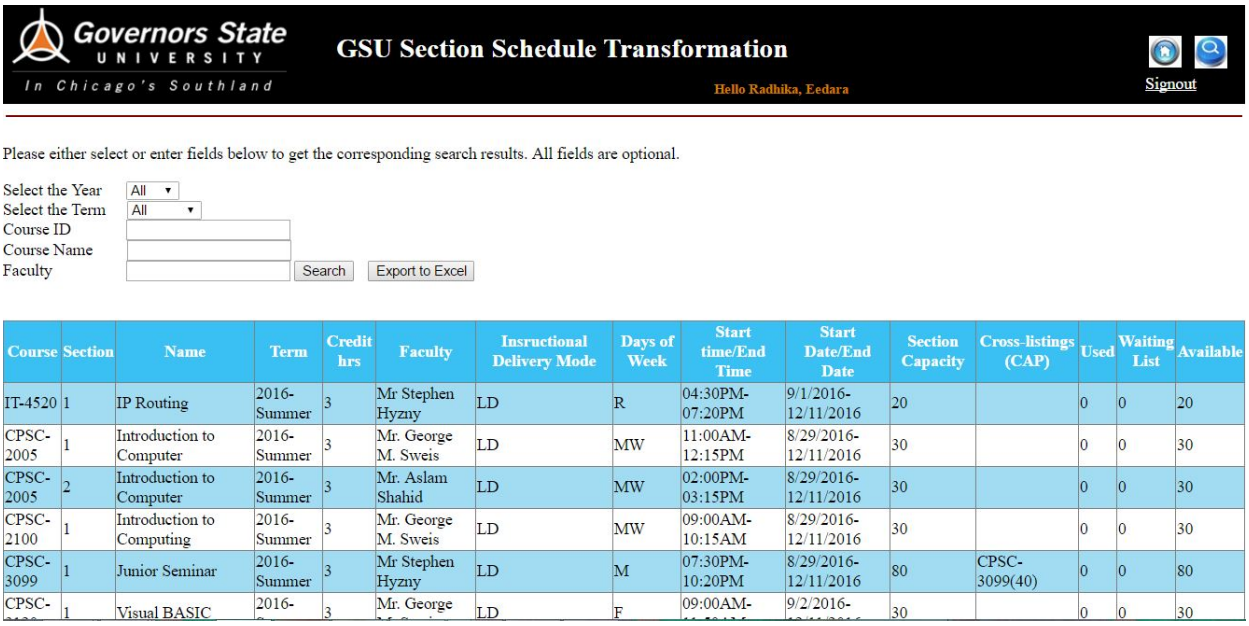


Fig. 13 – Search All

2.5 Capabilities

Support crew of this application needs knowledge on asp.net web forms, C# language and SQL server. Asp.net web forms and C# knowledge is needed to change any of the pages and business logic. Database SQL server knowledge is needed for user maintenance, taking backups and creating maintenance plans.

3 Project Requirements

3.1 Identification of Requirements

a. Current availability across multiple sections
b. To see which sections are filling up fast
c. Waiting list details of different sections
d. Planning sections for any student by days of the week.

3.2 Security and Fraud Prevention

Only authenticated users can login to the application and make any changes or use the application. There is no chance of any fraud activities on the application since the application can be used only by authenticated users also the database is secured by username and password. The data provided in text file is open to everyone from the website and hence the chances for loss or theft of data is least. Continuous monitoring is also done to identify any changes in the database and the application.

3.3 Release and Transition Plan

The GSU Section Schedule Transformation application is now deployed in local IIS server and executing on local machine.

4 *Project Design Description*

4.1 *Application Design*

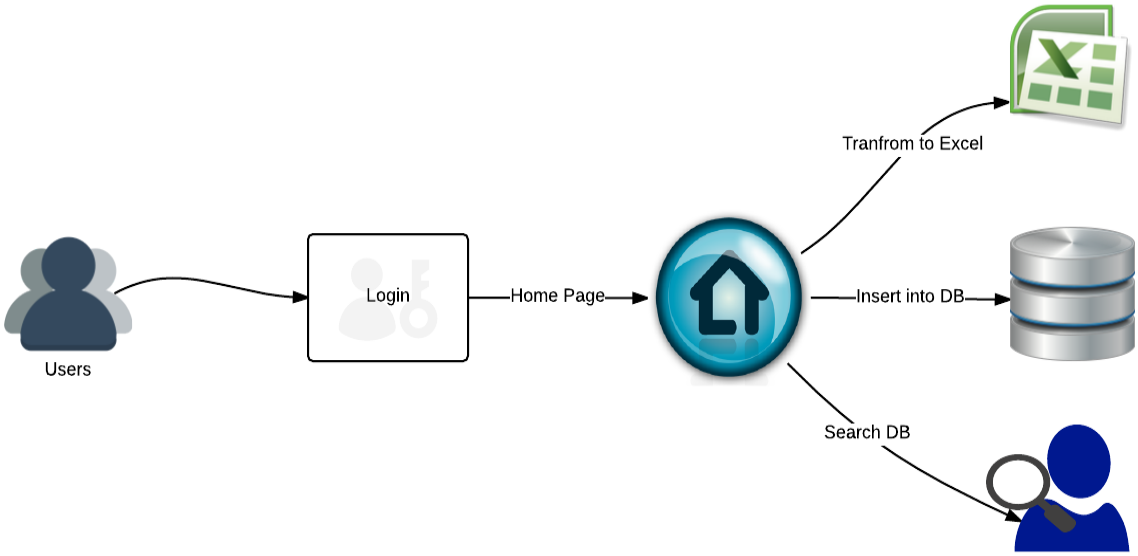


Fig. 14 Application User Interface

4.2 ***System Files***

.cs files

- *Default.aspx.cs*
- *Exceldatastore.aspx.cs*
- *Login.aspx.cs*
- *Search.aspx.cs*
- *Hearder.ascx.cs*
- *Footer.ascx.cs*

.aspx files

- *Default.aspx*
- *Exceldatastore.aspx*
- *Login.aspx*
- *Search.aspx*

CSS files

- Style.css

Name spaces

- *System.Collections.Generic*
- *System.Linq*
- *System.Data.OleDb*
- *System.Data.SqlClient*
- *System.Web*
- *System.Web.UI*
- *System.Web.UI.WebControls*
- *System.Text*
- *System.IO*
- *System.Text.RegularExpressions*

Bin files

- *Interop.Microsoft.Office.Core.dll*
- *Microsoft.Office.Interop.Excel.dll*
- *Microsoft.Vbe.Interop.dll*
- *office.dll*

4.3 ***Default.aspx code:***

```
//namespace GSU_Schedule
//{
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Data.OleDb;
    using System.Data.SqlClient;
    using System.Web;
    using System.Web.UI;
    using System.Web.UI.WebControls;
    using System.Text;
    using System.IO;
    using System.Text.RegularExpressions;

    using Excel =
Microsoft.Office.Interop.Excel;
using Microsoft.Office.Interop.Excel;

public partial class _Default :
System.Web.UI.Page
{
    //protected CCUtility Utility;

    protected void Page_Load(object
sender, EventArgs e)
    {
        //Utility = new CCUtility(this);
        if (Session["UserId"] == null)
        {

Response.Redirect("login.aspx");
        }
    }
    protected void
loadButton_Click(object sender, EventArgs
e)
    {
        if ((TextBox1.Text == "") ||
(DropDownList1.SelectedValue == "Not
Selected"))
        {
            Label2.Text = "Please enter
the Year and term";
            return;
        }
        else if(TextBox1.Text.Length != 4
|| !(TextBox1.Text.All(Char.IsDigit)))
        {
            Label2.Text = "Year is not in
the correct format";
            return;

        }
            else{
            Label2.Text = ""; }
        try
        {
            string[] lines =
System.IO.File.ReadAllLines(Server.MapPat
h("GSU_Schedule.txt"));

           foreach (string line in lines)
            {
                //string[] words =
Regex.Split(line, " ");
                string[] words =
Regex.Split(line, @"\s\s");
                foreach (var element in
words)
                {

                    if (element !=
String.Empty) infoTextBox.Text += element
+ "\r\n";
                }

            }

        }
        catch (Exception ex)
        {
            infoTextBox.Text = "The file
could not be read:";
            infoTextBox.Text =
ex.Message;
        }

    }
    protected void
excelButton_Click(object sender,
EventArgs e)
    {

        Excel.Application oXL = new
Excel.Application();
        Excel.Workbook oWB =
oXL.Workbooks.Open(Server.MapPath("Copy_s
chedule1.xlsx"));
        Excel.Sheets excelWorkSheet =
oWB.Sheets;

        Excel.Worksheet oWS =
oWB.Worksheets[1] as Excel.Worksheet;
        Excel.Range myRange =
oWS.UsedRange;


//Microsoft.Office.Interop.Excel.Range
cel = (Range)oXL.Cells[2, 18];
        //cel.Delete();

        Excel.Range excelRange =
oWS.UsedRange;
```

```
        string[] lines =
System.IO.File.ReadAllLines(Server.MapPat
h("GSU_Schedule.txt"));
        //int rowExcel = 2;
        int rownum = 1;
        int colnum = 1;

        foreach (string line in lines)
        {
            if (line.Length > 2)
            {
                string newrow =
line.Substring(0, 2);

                if (newrow == "CP" ||
newrow == "IT")
                {
                    rownum = rownum + 1;
                    colnum = 1;
                }
                else if (colnum == 1)
                { continue; }
            }
            string[] words;
            string b = line.Trim();
            if (b.Contains("PRIMARY") ||
b.Contains("SECONDARY"))
            { words = Regex.Split(line,
@"abc"); }
            else
            {
                //string[] words =
Regex.Split(line, "\t");
                words = Regex.Split(line,
@"\s\s");
            }
            int coltotal = words.Count();

            foreach (var element in
words)
            {
                if (element !=
String.Empty)
                {
                    if ((element.Length >
2) && (colnum == 1))
                    {
                        string coursecd =
element.Substring(0, 2);
                        if (coursecd ==
"CP")
                        {

infoTextBox.Text += element.Substring(0,
9) + "\r\n";

oWS.Cells[rownum, colnum] =
element.Substring(0, 9).Trim();
                                colnum += 1;

infoTextBox.Text += element.Substring(10,
2) + "\r\n";

oWS.Cells[rownum, colnum] =
element.Substring(10, 2).Trim();
                                colnum += 1;

infoTextBox.Text += TextBox1.Text.Trim()
+ "\r\n";

oWS.Cells[rownum, colnum] =
TextBox1.Text.Trim();
                                colnum += 1;

infoTextBox.Text +=
DropDownList1.SelectedValue.Trim() +
"\r\n";

oWS.Cells[rownum, colnum] =
DropDownList1.SelectedValue.Trim();
                                colnum += 1;

                        }
                        else if (coursecd
== "IT")
                        {

infoTextBox.Text += element.Substring(0,
7) + "\r\n";

oWS.Cells[rownum, colnum] =
element.Substring(0, 7).Trim();
                                colnum += 1;

infoTextBox.Text += element.Substring(8,
2) + "\r\n";

oWS.Cells[rownum, colnum] =
element.Substring(8, 2).Trim();
                                colnum += 1;

infoTextBox.Text += TextBox1.Text.Trim()
+ "\r\n";

oWS.Cells[rownum, colnum] =
TextBox1.Text.Trim();
                                colnum += 1;

infoTextBox.Text +=
DropDownList1.SelectedValue.Trim() +
"\r\n";
```

```
oWS.Cells[rownum, colnum] =
DropDownList1.SelectedValue.Trim();
                            colnum += 1;
                        }
                    }
                    else if
((element.Length > 2) && ((colnum == 7)
|| (colnum == 17)))
                    {
                        string a =
element.Trim();
                        if (a.Length ==
16)
                        {


oWS.Cells[rownum, colnum] =
a.Substring(0, 7);
                            colnum += 1;


oWS.Cells[rownum, colnum] =
a.Substring(8, 8);
                            colnum += 1;
                        }
                        else
                        {

oWS.Cells[rownum, colnum] = "NA";
                            colnum += 1;

oWS.Cells[rownum, colnum] = a;
                            colnum += 1;
                        }
                    }
                    else if
((element.Length > 2) && (colnum > 18))
                    {
                        string a =
element.Trim();


if (a.Contains("SECONDARY") &&
!a.Contains("take"))
                        {
                            int index =
a.IndexOf("SECONDARY");
                            string
coursesec = a.Substring(index+9,
15).Trim();
                            int
sectionindex = coursesec.Length - 3;
                            string course
= coursesec.Substring(0, sectionindex);
                            int
availindex = a.Length - 2;
                            string avail
= a.Substring(availindex, 2);
                            var cell =
(string)(oWS.Cells[rownum, colnum] as
Excel.Range).Value;

                            if (cell !=
null)
                            {
oWS.Cells[rownum, colnum] = cell + "\n" +
course + "(" + avail + ")"; }
                            else
                            {
oWS.Cells[rownum, colnum] = course + "("
+ avail + ")"; }
                        }

                    }
                    else
                    {

                        infoTextBox.Text
+= element + "\r\n";

                        oWS.Cells[rownum,
colnum] = element.Trim();     //
                        // }
                        colnum += 1;
                    }
                }

            }

        }



oWB.SaveCopyAs("C:/Users/RADHIKA/Document
s/Visual Studio
2012/WebSites/GSU_Schedule_File_Transform
ation_Tools/GSU_SChedule.xlsx");
        //oWB.Save();
        oWB.Close(false);
        oXL.Quit();
        Label2.Text = "Text file has been
successfully transformed to excel. <a
href='GSU_Schedule.xlsx'
Title='Transformed excel'>Click here</a>
to download";

    }

        protected void
btDataBase_Click(object sender, EventArgs
e)
    {
```

```
        if ((TextBox1.Text == "") ||
(DropDownList1.SelectedValue == "Not
Selected"))
        {
            Label2.Text = "Please enter
the Year and term";
            return;
        }
        else if (TextBox1.Text.Length !=
4 || !(TextBox1.Text.All(Char.IsDigit)))
        {
            Label2.Text = "Year is not in
the correct format";
            return;

        }
        else
        {
            Label2.Text = "";
        }

Response.Redirect("ExcelDataStore.aspx?yi
d=" + TextBox1.Text + "&tid=" +
DropDownList1.SelectedValue);
    }
        protected void
btSearch_Database_Click(object sender,
EventArgs e)
        {

Response.Redirect("Search.aspx");
        }
}
//}
```

4.4 ExcelDataStore Code:

```
namespace GSU_Schedule
{


    using System;
    using System.Collections;
    using System.ComponentModel;
    using System.Data;
    using System.Data.OleDb;
    using System.Drawing;
    using System.Web;
    using System.Web.SessionState;
    using System.Web.UI;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;
    using System.Data.SqlClient;


    public class PositionData
    {
        private string name;
        private string CatID;

        public PositionData(string Name,
string CategoryID)
        {
            this.name = Name;
            this.CatID = CategoryID;
        }

        public string Name
        {
            get
            {
                return name;
            }
        }

        public string CategoryID
        {
            get
            {
                return CatID;
            }
        }
    }

    public enum FieldTypes { Text,
Number, Date, Memo }

    public class CCUtility
    {

        protected HttpSessionState
Session;
        protected HttpServerUtility
Server;
        protected HttpRequest Request;
        protected HttpResponse Response;


        public static string ToSQL(string
Param, FieldTypes Type)
        {
            if (Param == null ||
Param.Length == 0)
            {
                return "Null";
            }
            else
            {
                string str =
Quote(Param);
                if (Type ==
FieldTypes.Number)
                {
                    return
str.Replace(',', '.');
                }
```

```
                else
                {
                    return "\'" + str +
"\'";
                }
            }
        }

        public CCUtility(object parent)
        {
            Session =
HttpContext.Current.Session;
            Server =
HttpContext.Current.Server;
            Request =
HttpContext.Current.Request;
            Response =
HttpContext.Current.Response;
            DBOpen();
        }

        public static String
GetValFromLOV(String val, String[] arr)
        {
            String ret = "";
            if (arr.Length % 2 == 0)
            {
                int temp =
Array.IndexOf(arr, val);
                ret = temp == -1 ? "" :
arr[temp + 1];
            }
            return ret;
        }

        public bool IsNumeric(object
source, string value)
        {
            try
            {
                Decimal temp =
Convert.ToDecimal(value);
                return true;
            }
            catch
            {
                return false;
            }
        }

        public static string Quote(string
Param)
        {
            if (Param == null ||
Param.Length == 0)
            {
                return "";
            }
            else
            {
                return Param.Replace("'",
"''");
            }
        }

        public static string
GetValue(DataRow row, string field)
        {
            if (row[field].ToString() ==
null)
                return "";
            else
                return
row[field].ToString();
        }

        public SqlConnection Connection;

        public DataSet FillDataSet(string
sSQL)
        {
            DataSet ds = new DataSet();
            SqlDataAdapter command = new
SqlDataAdapter(sSQL, Connection);
            return ds;
        }

        public int FillDataSet(string
sSQL, ref DataSet ds)
        {
            SqlDataAdapter command = new
SqlDataAdapter(sSQL, Connection);
            return command.Fill(ds,
"Table");
        }

        public int FillDataSet(string
sSQL, ref DataSet ds, int start, int
count)
        {
            SqlDataAdapter command = new
SqlDataAdapter(sSQL, Connection);
            return command.Fill(ds,
start, count, "Table");
        }


        public void DBOpen()
        {

           String sConnectionString =
System.Configuration.ConfigurationManager
.AppSettings["GSU"];
```

```
            Connection = new
SqlConnection(sConnectionString);
            Connection.Open();

        }

        public void DBClose()
        {
            Connection.Close();
        }

        public string GetParam(string
ParamName)
        {
            string Param =
Request.QueryString[ParamName];
            if (Param == null)
                Param =
Request.Form[ParamName];
            if (Param == null)
                return "";
            else
                return Param;
        }

        public string Dlookup(string
table, string field, string sWhere)
        {
            string sSQL = "SELECT " +
field + " FROM " + table + " WHERE " +
sWhere;

            SqlCommand command = new
SqlCommand(sSQL, Connection);
            SqlDataReader reader =
command.ExecuteReader(CommandBehavior.Sin
gleRow);
            string sReturn;

            if (reader.Read())
            {
                sReturn =
reader[0].ToString();
                if (sReturn == null)
                    sReturn = "";
            }
            else
            {
                sReturn = "";
            }

            reader.Close();
            return sReturn;
        }

        public int DlookupInt(string
table, string field, string sWhere)
        {
            string sSQL = "SELECT " +
field + " FROM " + table + " WHERE " +
sWhere;

            SqlCommand command = new
SqlCommand(sSQL, Connection);
            SqlDataReader reader =
command.ExecuteReader(CommandBehavior.Sin
gleRow);
            int iReturn = -1;

            if (reader.Read())
            {
                iReturn =
reader.GetInt32(0);
            }

            reader.Close();
            return iReturn;
        }

        public void Execute(string sSQL)
        {
            SqlCommand cmd = new
SqlCommand(sSQL, Connection);
            cmd.ExecuteNonQuery();
        }

        public void
buildListBox(ListItemCollection Cars,
string sSQL, string sId, string sTitle,
string CustomInitialDisplayValue, string
CustomInitialSubmitValue)
        {
            Cars.Clear();
            String s =
System.Configuration.ConfigurationManager
.AppSettings["GSU"];
            SqlConnection Conn = new
SqlConnection(s);
            Conn.Open();

                    SqlCommand command =
new SqlCommand(sSQL, Conn);
            SqlDataReader reader =
command.ExecuteReader(CommandBehavior.Clo
seConnection);


            if (CustomInitialDisplayValue
!= null) Cars.Add(new
ListItem(CustomInitialDisplayValue,
CustomInitialSubmitValue));

            while (reader.Read())
            {
                if (sId == "" && sTitle
== "")
```

```
                {
                    Cars.Add(new
ListItem(reader[1].ToString(),
reader[0].ToString()));
                }
                else
                {
                    Cars.Add(new
ListItem(reader[sTitle].ToString(),
reader[sId].ToString()));
                }
            }
            reader.Close();
            Conn.Close();
        }

        public void
buildListBox(ListItemCollection Cars,
string[] values, string
CustomInitialDisplayValue, string
CustomInitialSubmitValue)
        {
            Cars.Clear();
            if (CustomInitialDisplayValue
!= null) Cars.Add(new
ListItem(CustomInitialDisplayValue,
CustomInitialSubmitValue));
            for (int i = 0; i <
values.Length; i += 2) Cars.Add(new
ListItem(values[i + 1], values[i]));
        }

        public ICollection
buildListBox(string sSQL, string sId,
string sTitle, string
CustomInitialDisplayValue, string
CustomInitialSubmitValue)
        {
            DataRow row;

            SqlDataAdapter command = new
SqlDataAdapter(sSQL, Connection);
            DataSet ds = new DataSet();
            ds.Tables.Add("lookup");

            DataColumn column = new
DataColumn();
            column.DataType =
System.Type.GetType("System.String");
            column.ColumnName = sId;

ds.Tables[0].Columns.Add(column);

            column = new DataColumn();
            column.DataType =
System.Type.GetType("System.String");
            column.ColumnName = sTitle;
ds.Tables[0].Columns.Add(column);

            if (CustomInitialDisplayValue
!= null)
            {
                row =
ds.Tables[0].NewRow();
                row[0] =
CustomInitialSubmitValue;
                row[1] =
CustomInitialDisplayValue;

ds.Tables[0].Rows.Add(row);
            }

            command.Fill(ds, "lookup");
            return new
DataView(ds.Tables[0]);
        }

        public static string
getCheckBoxValue(string sVal, string
CheckedValue, string UnCheckedValue,
FieldTypes Type)
        {
            if (sVal.Length == 0)
            {
                return
ToSQL(UnCheckedValue, Type);
            }
            else
            {
                return
ToSQL(CheckedValue, Type);
            }
        }

        public void CheckSecurity(int
iLevel)
        {
            if (Session["UserID"] == null
|| Session["UserID"].ToString().Length ==
0)
            {

Response.Redirect("Login.aspx?QueryString
=" +
Server.UrlEncode(Request.ServerVariables[
"QUERY_STRING"]) + "&ret_page=" +
Server.UrlEncode(Request.ServerVariables[
"SCRIPT_NAME"]));
            }
            else
            {
                if
(Int16.Parse(Session["UserRights"].ToStri
ng()) < iLevel)
```

```
Response.Redirect("Login.aspx?QueryString
=" +
Server.UrlEncode(Request.ServerVariables[
"QUERY_STRING"]) + "&ret_page=" +
Server.UrlEncode(Request.ServerVariables[
"SCRIPT_NAME"]));
            }
        }
    }
```

4.5 Search Page:

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Search : System.Web.UI.Page
{
   protected void Page_Load(object sender,
EventArgs e)
   {
      if (Session["UserId"] == null)
      {
         Response.Redirect("login.aspx");
      }
      Export_Excel.Visible = false;
   }

   protected void Search_button_Click(object sender,
EventArgs e)
   {
      GetData();
      Export_Excel.Visible = true;
   }
   protected void grd_PageIndexChanging(object
sender, GridViewPageEventArgs e)
   {
      GetData();
      grd.PageIndex = e.NewPageIndex;
      grd.DataBind();

   }

   private void GetData()
   {
      string connectionString =
System.Configuration.ConfigurationManager.AppSet
tings["GSU"];
      //string a = Search_name.Text;
      //
      // In a using statement, acquire the
SqlConnection as a resource.
      //
      using (SqlConnection con = new
SqlConnection(connectionString))
      {
         //
         // Open the SqlConnection.
         //
         con.Open();
         //
         // The following code uses an SqlCommand
based on the SqlConnection.
         //
         DataSet ds = new DataSet();
         using (SqlCommand command = new
SqlCommand("SELECT
[Course],[Section],[Name],[Year]+'-'+[Term] as
[Term],[Credit hrs],[Faculty],[Insructional Delivery
Mode],[Days of Week],[Start Time] + '-' + [End
Time] as [Start time/End Time] ,left([Start
Date],len([Start Date]) -12) +'-'+left([End
Date],len([End Date]) -12) as [Start Date/End
Date],[Section Capacity],[Cross-listings
(CAP)],[Used],[Waiting List],[Available] FROM
[GSU_Term] where ([Year] = '" +
DropDownList1.SelectedValue + "' or '" +
DropDownList1.SelectedValue + "' = 'All') and
([Term] = '" + DropDownList2.SelectedValue + "' or
'" + DropDownList2.SelectedValue + "' = 'All') and
([Course] like '%" + CourseID.Text + "%' or '" +
CourseID.Text + "' =") and ([Faculty] like '%" +
Search_name.Text + "%' or '" + Search_name.Text +
"' =") and ([Name] like '%" + CourseName.Text +
"%' or '" + CourseName.Text + "' =")", con))
         {
            SqlDataAdapter adp = new
SqlDataAdapter(command);
            adp.Fill(ds);

            if (ds.Tables[0].Rows.Count > 0)
            {
               grd.DataSource = ds;
               grd.DataBind();
               Export_Excel.Visible = true;
            }
            else
            {
               grd.DataSource = null;
               grd.DataBind();
               Export_Excel.Visible = false;
            }
         }
      }
```

```
    }
    protected void
DropDownList1_SelectedIndexChanged(object
sender, EventArgs e)
    {

    }
    protected void Export_Excel_button_Click(object
sender, EventArgs e)
    {
        Response.Clear();
        Response.Buffer = true;
        Response.AddHeader("content-disposition",
"attachment;filename=GridViewExport.xlsx");
        Response.Charset = "";
        Response.ContentType = "application/vnd.ms-
excel";
        using (StringWriter sw = new StringWriter())
        {
            HtmlTextWriter hw = new
HtmlTextWriter(sw);
            //To Export all pages
            grd.AllowPaging = false;
            this.GetData();

            grd.HeaderRow.BackColor = Color.White;
            foreach (TableCell cell in
grd.HeaderRow.Cells)
            {
                cell.BackColor =
grd.HeaderStyle.BackColor;
            }
            foreach (GridViewRow row in grd.Rows)
            {
                row.BackColor = Color.White;
                foreach (TableCell cell in row.Cells)
                {
                    if (row.RowIndex % 2 == 0)
                    {
                        cell.BackColor =
grd.AlternatingRowStyle.BackColor;
                    }
                    else
                    {
                        cell.BackColor =
grd.RowStyle.BackColor;
                    }
                    cell.CssClass = "textmode";
                }
            }

            grd.RenderControl(hw);

            //style to format numbers to string
            string style = @"<style> .textmode { }
</style>";
            Response.Write(style);
            Response.Output.Write(sw.ToString());
            Response.Flush();
            Response.End();
        }
    }
    public override void
VerifyRenderingInServerForm(Control control)
    {
        /* Verifies that the control is rendered */
    }
}
```

4.6 Database Scripts

Member Table

```
USE [GSU_Schedule]
GO

/****** Object:  Table [dbo].[members]
Script Date: 5/2/2016 11:15:19 PM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_PADDING ON
GO

CREATE TABLE [dbo].[members](
        [member_id] [int] IDENTITY(1,1) NOT
NULL,
        [member_login] [varchar](20) NOT
NULL,
        [member_password] [varchar](20) NOT
NULL,
        [member_level] [int] NOT NULL
DEFAULT ((1)),
        [first_name] [varchar](50) NOT NULL,
        [last_name] [varchar](50) NOT NULL,
        [email] [varchar](50) NOT NULL,
        [phone] [varchar](50) NOT NULL,
        [address] [varchar](250) NULL,
        [notes] [text] NULL,
PRIMARY KEY CLUSTERED
(
        [member_id] ASC
)WITH (PAD_INDEX = OFF,
STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
```

```
SET ANSI_PADDING OFF
GO
```

GSU_Term:

```
USE [GSU_Schedule]
GO

/****** Object:  Table [dbo].[GSU_Term]
Script Date: 5/2/2016 11:15:08 PM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[GSU_Term](
	[Course] [nvarchar](50) NULL,
	[Section] [nvarchar](50) NULL,
	[Year] [nvarchar](50) NULL,
	[Term] [nvarchar](50) NULL,
	[Name] [nvarchar](max) NULL,
	[Days of Week] [nvarchar](50) NULL,
	[Start Time] [nvarchar](50) NULL,
	[Start Date] [nvarchar](50) NULL,
	[Credit hrs] [smallint] NULL,
	[Section Minimum] [smallint] NULL,
	[Section capacity] [smallint] NULL,
	[Used] [smallint] NULL,
	[Waiting List] [smallint] NULL,
	[Available] [smallint] NULL,
	[Faculty] [nvarchar](max) NULL,
	[Insructional Delivery Mode]
[nchar](10) NULL,
	[End Time] [nvarchar](50) NULL,
	[End Date] [nvarchar](50) NULL,
	[Cross-listings (CAP)]
[nvarchar](max) NULL,
	[member_id] [int] NULL,
	[uploadeddate] [datetime] NOT NULL
CONSTRAINT [DF_GSU_Term_uploadeddate]
DEFAULT (getdate())
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO

ALTER TABLE [dbo].[GSU_Term]  WITH NOCHECK
ADD  CONSTRAINT [FK_term_member_id] FOREIGN
KEY([member_id])
REFERENCES [dbo].[members] ([member_id])
GO

ALTER TABLE [dbo].[GSU_Term] CHECK
CONSTRAINT [FK_term_member_id]
GO
```

Instructor Table

```
USE [GSU_Schedule]
GO

/****** Object:  Table [dbo].[Instructor]
Script Date: 5/2/2016 11:15:15 PM ******/
SET ANSI_NULLS ON
GOa

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Instructor](
	[InstructorID] [int] NOT NULL,
	[Name] [nvarchar](max) NULL,
	[Room] [nvarchar](50) NULL,
	[Phone] [nvarchar](12) NULL,
	[email] [nvarchar](250) NULL,
	[member_id] [int] NULL,
 CONSTRAINT [PK_Instructor] PRIMARY KEY
CLUSTERED
(
	[InstructorID] ASC
)WITH (PAD_INDEX = OFF,
STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO

ALTER TABLE [dbo].[Instructor]  WITH CHECK
ADD  CONSTRAINT [FK_Instructor_members]
FOREIGN KEY([member_id])
REFERENCES [dbo].[members] ([member_id])
GO

ALTER TABLE [dbo].[Instructor] CHECK
CONSTRAINT [FK_Instructor_members]
GO
```

Course Table Code:

```
USE [GSU_Schedule]
GO

/****** Object:  Table [dbo].[Course]
Script Date: 5/2/2016 11:15:03 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Course](
	[CourseID] [int] NOT NULL,
	[Course] [nvarchar](50) NOT NULL,
	[CourseName] [nvarchar](max) NOT
NULL,
	[OnCampus] [nvarchar](1) NULL,
	[Online] [nvarchar](1) NULL,
	[member_id] [int] NULL,
 CONSTRAINT [PK_Course] PRIMARY KEY
CLUSTERED
(
	[CourseID] ASC
```

```
)WITH (PAD_INDEX = OFF,
STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
ALTER TABLE [dbo].[Course]  WITH CHECK ADD
CONSTRAINT [FK_Course_Course] FOREIGN
KEY([member_id])
REFERENCES [dbo].[members] ([member_id])
GO
ALTER TABLE [dbo].[Course] CHECK CONSTRAINT
[FK_Course_Course]
GO
```

Teach table:

```
USE [GSU_Schedule]
GO

/****** Object:  Table [dbo].[Teach]
Script Date: 5/2/2016 11:15:24 PM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Teach](
       [TeachID] [int] NOT NULL,
       [Year] [nvarchar](50) NULL,
       [Term] [nvarchar](50) NULL,


       [InstructorID] [int] NULL,
       [courseID] [int] NULL,
       [Section] [nvarchar](50) NULL,
       [member_id] [int] NULL,
 CONSTRAINT [PK_Teach] PRIMARY KEY
CLUSTERED
(
       [TeachID] ASC
)WITH (PAD_INDEX = OFF,
STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Teach]  WITH CHECK ADD
CONSTRAINT [FK_Teach_Course1] FOREIGN
KEY([courseID])
REFERENCES [dbo].[Course] ([CourseID])
GO
ALTER TABLE [dbo].[Teach] CHECK CONSTRAINT
[FK_Teach_Course1]
GO
ALTER TABLE [dbo].[Teach]  WITH NOCHECK ADD
CONSTRAINT [FK_Teach_Instructor] FOREIGN
KEY([InstructorID])
REFERENCES [dbo].[Instructor]
([InstructorID])
GO
ALTER TABLE [dbo].[Teach] CHECK CONSTRAINT
[FK_Teach_Instructor]
GO
ALTER TABLE [dbo].[Teach]  WITH CHECK ADD
CONSTRAINT [FK_Teach_members] FOREIGN
KEY([member_id])
REFERENCES [dbo].[members] ([member_id])
GO

ALTER TABLE [dbo].[Teach] CHECK CONSTRAINT
[FK_Teach_members]
GO
```

5 Internal/external Interface Impacts and Specification

- This application focuses on the data structure that is available in my-gsu website.
- The downloaded text file has lot of special errors on the saved text file that are transformed to unique spacing and format.
- The redesign of the text file should be in standard format. Any changes in the text file will impact the whole application that lead to error on the page.
- To overcome these impacts, the design of the data in the text file must be re-oriented in such a way that fits the application space.
- Externally the downloaded file must be stored in a specific folder for the file to be uploaded in the database. This impact is fixed by adding the source folder location to the source code.

6 Project Design Units Impacts

- **Login:** Registered users are able to access the web application.
- **Search:** User can search the existing data from the database. User can select the term or faculty name and other available options from the application.
- **ETL:** User has to enter the term, accordingly the text file will be loaded in the Text box and the user can transform the text file into Excel. The file transformation is done using existing text file from the database or from the local machine.
- **Insert:** The transformed data is then uploaded into the database using 'insert into database' button. Then the file is updated in the database that can be downloaded later or view from the database.

7 System Requirements

7.1 Hardware Requirements

Visual Studio 2012

- 1.6 GHz or faster processor.
- 1 GB of RAM (1.5 GB if running on a virtual machine)
- 10 GB (NTFS) of available hard disk space.
- 5400 RPM hard drive.
- DirectX 9-capable video card running at 1024 x 768 or higher display resolution.

SQL Server Database

- 6 GB Hard drive space
- Installation discs
- Monitor
- Internet

7.2 Software Requirements

- Visual Studio 2012
- .NET Framework 4.0
- Microsoft SQL Server Management Studio 2012
- Windows 7 or higher
- Internet explorer or google chrome browser.

8 Acknowledgements

I would first like to thank my thesis advisor Dr. Soon O Park of the Computer Science Department at Governors State University. The door to Dr. Park office was always open whenever I ran into a trouble spot or had a question about my project. She consistently allowed this project to be my own work, but steered me in the right the direction whenever she thought I needed it.

I would also like to thank the experts who were involved directly or indirectly in the validation survey for this project. Without their passionate participation and input, the validation survey could not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents, my siblings and my Dear friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

9 References

1. Art Gittleman (2012). Computing with C# and the .NET Framework. Available from https://books.google.com/books?isbn=1449615503
2. Microsoft Developer Network. Retrieved March 28, 2016 from the Microsoft Website: https://msdn.microsoft.com/en-us/library/8yttk7sy(v=vs.110).aspx
3. Raps MK. (2016). ASP.NET Grid View. Retrieved from http://asp.net-informations.com/gridview/asp-gridview.htm
4. Mudassar Ahmed Khan. (May 11, 2013). How to export Grid View to Excel file in ASP.Net with formatting and styles using C# and VB.Net. Retrieved from http://www.aspsnippets.com/Articles/Export-GridView-to-Excel-in-ASPNet-with-Formatting-using-C-and-VBNet.aspx
5. Uploading an Excel sheet and importing the data into SQL Server database. (May 04, 2012). Message posted to http://stackoverflow.com/questions/10447015/uploading-an-excel-sheet-and-importing-the-data-into-sql-server-database
6. Aero Host. (2013). ASPX and ASP.NET. Retrieved from http://aerohost.com/dotnet-aspx.htm