****

**PROJECT PROPOSAL**

**Name of the Project-**

**Cafeteria Vernier (CV)**

**Objective:**

1. CV (Cafeteria Vernier) is found in almost all cyber café and gaming zone.

2. Managing customer accounts in efficient way.

3. At the same time recording customer activity.

**Details of (CV):**

CV provides a fast and efficient way of dealing with customers. Traditionally in different commercial institutions like Cyber café or Gaming zone they deal with the time the customer use their services and ask money for the amount of time the customer use. They do not keep a track of the customer information and thus customer do not get any further benefits by using the same institution time and again as they have to give the customer information again and again. This is time consuming and economically non beneficial. Cafeteria Vernier does this and a lot more. This has the ability to keep customer information and a customer gets the facility to maintain separate account and thus buy time and use that whenever he want in a one minute pulse rate. It also helps the administrator to maintain the system smoothly and gain proper economic benefits by this it saves a lot of time for management. For example- traditionally a customer if uses the service of an institution for forty minutes he has to pay either for thirty minutes or one hour which may create loss for the institution or the customer. But CV gives the chance to pay per minute and maintain an account which creates proper management.

**Submitted To:** **Prof. Dr.Md.Humayun Kabir**

STATE UNIVERSITY OF BANGLADESH

(*Lecturer of CSE Department )*

**Submitted By:**

* Md.Hasanuzzaman

UG02-24-10-013

Deadline 9th August

**Scope of the System**

Cafeteria Vernier (CV) can become increasingly common across the commercial sector due to its increased functionality. In order to function a CV system, a customer has to create an account at the admins or employee desk. Typically, customer has to provide the information like a username, a phone number, Email address, password, photo etc. Customer ID is necessary in identifying the user account from the database at the admin or employee desk. A team account can be made at the admin or employee desk by providing the data like a team name, Admin name, Adding team members, providing photo. A team ID is necessary in identifying the team account from the database at the admin or employee desk. An employee has to create an account at the admin desk by providing the information like a user name, a phone number, Email address, password, photo and has to select privileges which the employee will have the authority to analyze. An employee’s ID is necessary to identifying the team account from the database at the admin desk. A customer for enjoying the services of the institution has to recharge his or team account by selecting the customer ID or team ID and paying for selected minute or rate at the admin or employee desk. Recharge history and customer login history can be seen from the admin or employee desk for the better management of the system. Business summary can be seen at the admin or employee desk about the credit transition on daily basis. Email can be send from the admin or employee desk to customers for different notifications about the institutions. If any disorder or power fall occurs in the customer’s desk while the customer is logged in then the customer login status can be reset from the admin or employee desk. Database can be backed up or can be restored from the admin or employee desk in any kind of emergency.

A customer can login from a customer desk and enjoy his bought minutes. He can also change his password, change customer profile, view login history, view recharge history, create and update team and view team recharge history.

**Tools Used**

**Platform:** .NET 4.0.

**Languages:** C# 4.0 (Code Behind) and XAML (User Interface).

**Database:** SQL Server 2008 R2.

**Patten:** MVVM (Model-View-View Model)

**Technology`s**

1. WPF ( Windows Presentation Foundation)
2. WCF (Windows Commination Foundation)
3. Entity Framework (The “Model First” approach)
4. Linq (Languages Integrated Query)
5. Linq To Entity
6. Linq to Object
7. Linq to XML
8. Regex (Regular Expression)

**Software`s**

1. Visual Studio 2010 (10.0)
2. Expression Blend 4.0
3. MS Visio 2010
4. Adobe Photoshop CS 5

**1. Process Flow Diagram of Cafeteria vernier:**

1.1) Process Flow Diagram of server.

1.2) Process Flow Diagram of Service`s.

1.3) Process Flow Diagram of Client.

Process Flow Diagram of server.



Process Flow Diagram of Service`s.



Process Flow Diagram of Client



List of Actor:

1. Administrator or Employee
2. Customer

List of Use Case by the Actor:

1. Administrator or Employee:
2. Login
3. Create And Update Employee Account
4. Create And Update Customer Account
5. Create And Update Team Account
6. Account Recharge
7. Recharge History
8. Customer Login History
9. View Cash
10. Cash History
11. View Business Summary
12. Rate Setup
13. Customer Account Maintenance
14. Team Account Maintenance
15. Change Password
16. Screenshots
17. Send Email
18. Email Account Setup
19. General Setting
20. Counter`s Information
21. Database Backup and Restore
22. Customer Account status Reset

Use Case Narration:

* 1. **Use case 1: Login**

Primary Actor: Administrator or Employee

Stakeholder or Interests: Administrator or Employee

Precondition: Administrator or Employee ready to use for login and logout.

Post condition: Administrator or Employee logged in or logout.

Main Success Scenario:

1. Administrator or Employee type username and password.
2. Click on “Login” Button.
3. Load Administrator or Employee’s privileges.
4. Configuring software features based on Privileges.
   1. **Use case 2: Create And Update Employee Account**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: Add New or Update or Delete Employee.

Main Success Scenario:

1. Assign a Username. (Must be unique.)
2. Assign a Phone number. (Must be valid.)
3. Assign an Email address. (Must be valid.)
4. Provide a password.
5. Address may or may not be provided.
6. Provide a photo.
7. Select Privileges.
8. Click on “Update” button.
9. System adds new Employee and Display conformation messages.

Alternate Success Scenario:

1. If the Employee already exists then Display warning message and Update Employee information and Display conformation messages after update.
2. Click on “Delete” Button to delete a selected Employee. System Display warning messages before deleting and Display conformation messages after deleting.
   1. **Use case 3: Create and Update Customer Account**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: Add New or update or Delete Customer.

Main Success Scenario:

1. Assign a Username. (Must be unique.)
2. Assign a Phone number (Must be valid.)
3. Assign an Email address (Must be valid.)
4. Assign a Name (Must Assign.)
5. Provide a password.
6. Assign an Address. (May or may not be provided.)
7. Assign National ID. (May or may not be provided.)
8. Provide a photo.
9. Click on “Update” Button.
10. System adds new Customer and Display conformation messages.

Alternate Success Scenario:

1. If the customer already exists then Display a warning messages and Update customer information and Display confirmation messages after update.
2. Click on “Delete” Button to delete a selected Customer. System Display warning messages before deleting and Display confirmation messages after deleting.
   1. **Use case 4: Create and update Team Account**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: Add New or update or delete Team.

Main Success Scenario:

1. Assign a Team Name. (Must be unique.)
2. Assign an Admin Name. (Must have.)
3. Add Team members. (May or may not be added.)
4. Provide a photo.
5. Click on “Update” Button.
6. System adds new team and display confirmation messages.

Alternate Success Scenario:

1. If the team already exists then it can add and remove members.
2. Click on “Update” Button.
3. Display Warning messages
4. Finally Display confirmation messages after updating team information.
5. Click on “Delete” Button to delete a selected Team. System Display warning messages before deleting and Display confirmation messages after deleting.
   1. **Use case 5: Account Recharge**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: Update Customer or Team Account.

Main Success Scenario:

1. Select a Customer Name or Team Name.
2. Select minutes or Rate
3. Click on “Update” Button
4. Add minutes with existing account balance.
5. Display confirmation messages.
   1. **Use case 6: Recharge History**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: View Customer or Team Account Recharge History.

Main Success Scenario:

1. Select a Customer Name or Team Name.
2. Select any following search criteria.
3. Select By Name or For All.
4. By Date.
5. Between two Date.
6. Any Date.
7. Click on “Search” Button.
8. Display recharges history.

Alternative Success Scenario:

* 1. Delete History based on selected History or all.
  2. Display confirmation after deleting.
  3. **Use case 7: Customer Login History**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: View and Delete Customer Login History.

Main Success Scenario:

1. Select any following search criteria.
2. Select By Name or For All.
3. By Date.
4. Between two Date.
5. Any Date.
6. Click on “Search” Button
7. Display a list of Login History.

Alternative success scenario:

1. Delete History based on selected History or all.
2. Display confirmation messages after deleting.
   1. **Use case 8: View Cash**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: View and Update Cash.

Main Success Scenario:

1. Select a Date (Current Date selected by default).
2. Click on “Search” Button
3. View Cash based on selected date.

Alternative Success Scenario:

1. Edit and Update Cash.
2. Display confirmation messages after update cash.
   1. **Use case 9: Cash History**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: View Cash History.

Main Success Scenario:

1. Select any following search criteria.
2. By Date.
3. Between to Date.
4. By All.
5. Click on “Search” Button.
6. View a list of Cash information.

Alternative Success Scenario:

1. Making report based on search result.
2. Delete selected Cash information
3. Display confirmation messages after deleting.
   1. **Use case 10: View Business Summary**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: View business summary.

Main Success Scenario:

1. Select Date (Current Date selected by default).
2. Click on “Search” button.
3. View Cash, total sales minute and how many customers are logged in.
   1. **Use case 11: Rate Setup**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: Update or Insert Rate.

Main Success Scenario:

1. Provide minutes and it`s amount.
2. Click on “Update” button.
3. Display confirmation messages after insert.

Alternative Success Scenario:

1. If minutes or amount already exists then display a warning messages and update rate information and display confirmation messages after updating.
2. Delete selected rate information.
   1. **Use case 12: Customer Account Maintenance**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: View Customer Information`s.

Main Success Scenario:

1. Select any following search criteria.
2. By Name
3. By Date
4. Date is Equal.
5. Date is below.
6. Between two date.
7. By Minutes
8. Minutes are Equal.
9. Minutes are below.
10. Click on “Search” Button.
11. Display a list of Customer Account Information.

Alternative Success scenario:

1. Delete customer information based on selected customer or all customers.
   1. **Use case 13: Team Account Maintenance**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: View Team Information`s.

Main Success Scenario:

1. Select any following search criteria.
2. By Name
3. By Date
4. Date is Equal.
5. Date is below.
6. Between two date.
7. By Minutes
8. Minutes are Equal.
9. Minutes are below.
10. Click on “Search” Button.
11. Display a list of Team Information.

Alternative success scenario:

1. Delete team information based on selected team or all teams.
   1. **Use case 14: Change Password.**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: Change Employee or Administrator Password

Main Success Scenario:

1. Assign current password.
2. Assign new password.
3. Re-enter new password.
4. Click on “Update” Button.
5. Display confirmation messages.
   1. **Use case 15: Screenshot.**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: Taking screen snapshot after a particular time.

Main Success Scenario:

1. Provide a Time Span.
2. Click on “Start” button.

Alternative Success Scenario:

1. Click on “Stop” button to stop taking screenshot.
   1. **Use case 16: Send Email.**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: Send email to the customer’s.

Main Success Scenario:

1. Select any following email sending criteria.
2. Send email only One
3. Send email all.
4. Assign a subject.
5. Provide necessary information in email body.
6. Click on “Send” Button.
   1. **Use case 17: Email Account Setup.**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: Update sender email account information.

Main Success Scenario:

1. Assign a email address
2. Assign it`s password.
   1. **Use case 18: General Setting.**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: Update general settings.

Main Success Scenario:

1. Assign that software will start at computer’s start or not.
   1. **Use case 19. Counter`s Information.**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: View counter`s information.

Main Success Scenario:

1. View counter name.
2. View counter number
3. View username (if any customer is currently logged in )
4. View account balance (if any customer is currently logged in )
   1. **Use case 21: Database backup and restore.**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: Backup Database or Restore Database.

Main Success Scenario:

1. Database backup.
2. Assign a folder path.
3. Click on “Backup” Button.
4. Display confirmation messages after database backup.
5. Database Restore.
6. Select a .bak file.
7. Click on “Restore” Button.
8. Display confirmation messages after database restore.
   1. **Use case 22: Customer login status reset.**

Primary Actor: Administrator or Employee (Privileges Required).

Stakeholder or Interests: Administrator or Employee (Privileges Required).

Precondition: Login as Administrator or Privilege Required for Employee.

Post condition: Change login status if it shows still logged in after a customer logged out or may any other problem.

Main Success Scenario:

1. Select a customer.
2. Click on “Update” Button.
3. Display confirmation messages after updating.
4. Customer
5. Customer Login
6. Change password.
7. Change customer profile.
8. View login history
9. View recharge history
10. Create and update team
11. View team recharges history.

Use Case Narration:

* 1. **Use case 1: Customer login.**

Primary Actor: Customer.

Stakeholder or Interests: Customer.

Precondition: Customer logged in.

Post condition: Start decrementing balance from customer account.

Main Success Scenario:

1. Assign username.
2. Assign password.
3. Assign team name (if customer want to login in team)
4. Click on “Login” Button.
   1. **Use case 2: Change Password.**

Primary Actor: Customer.

Stakeholder or Interests: Customer.

Precondition: Customer logged in.

Post condition: Update customer login password.

Main Success Scenario:

1. Assign current password.
2. Assign new password.
3. Re-enter new password.
4. Click on “Update” Button.
5. Display confirmation messages.
   1. **Use case 3: Change customer profile.**

Primary Actor: Customer.

Stakeholder or Interests: Customer.

Precondition: Customer logged in.

Post condition: Update customer profile.

Main Success Scenario:

1. Change photo.
2. Change name.
3. Change phone.
4. Change email.
5. Change National ID.
6. Change address.
7. Click on “Update” Button.
8. Display confirmation messages.
   1. **Use case 4: View login history.**

Primary Actor: Customer.

Stakeholder or Interests: Customer.

Precondition: Customer logged in.

Post condition: View login history.

Main Success Scenario:

1. Select any following search criteria.
2. By Date.
3. Between to Date.
4. By All.
5. Click on “search” button.
6. Show a list of login information.
   1. **Use case 5: View recharges history.**

Primary Actor: Customer.

Stakeholder or Interests: Customer.

Precondition: Customer logged in.

Post condition: View recharges history.

Main Success Scenario:

1. Select any following search criteria.
2. By Date.
3. Between to Date.
4. By All.
5. Click on “Search” Button.
6. Show a list of recharge information.
   1. **Use case 6: Create and update team.**

Primary Actor: Customer.

Stakeholder or Interests: Customer.

Precondition: Customer logged in.

Post condition: Create new team or update old team.

Main Success Scenario:

1. Create new team.
   1. Assign a Team Name. (Must be unique.)
   2. Assign an Admin Name. (Must have.)
   3. Add Team members. (May or may not be added.)
   4. Provide a photo.
   5. Click on “Update” Button.
   6. Display confirmation messages.

Alternative Success Scenario:

1. Update old team (If the customer is a team administrator)
   1. Change photo
   2. Add and remove members.
   3. Click on “update” button.
   4. Display confirmation messages.
2. Delete a team.
   1. **Use case 7: View team recharges history.**

Primary Actor: Customer.

Stakeholder or Interests: Customer.

Precondition: Customer logged in.

Post condition: View team recharges history.

Main Success Scenario:

* 1. Select any following search criteria.
     1. By Date.
     2. Between to Date.
     3. By All.
  2. Click on “Search” Button.
  3. Show a list of team recharge information.

**Use-Case Diagram of CV**





Entity Relation Diagram of CV



Schema Diagram of CV

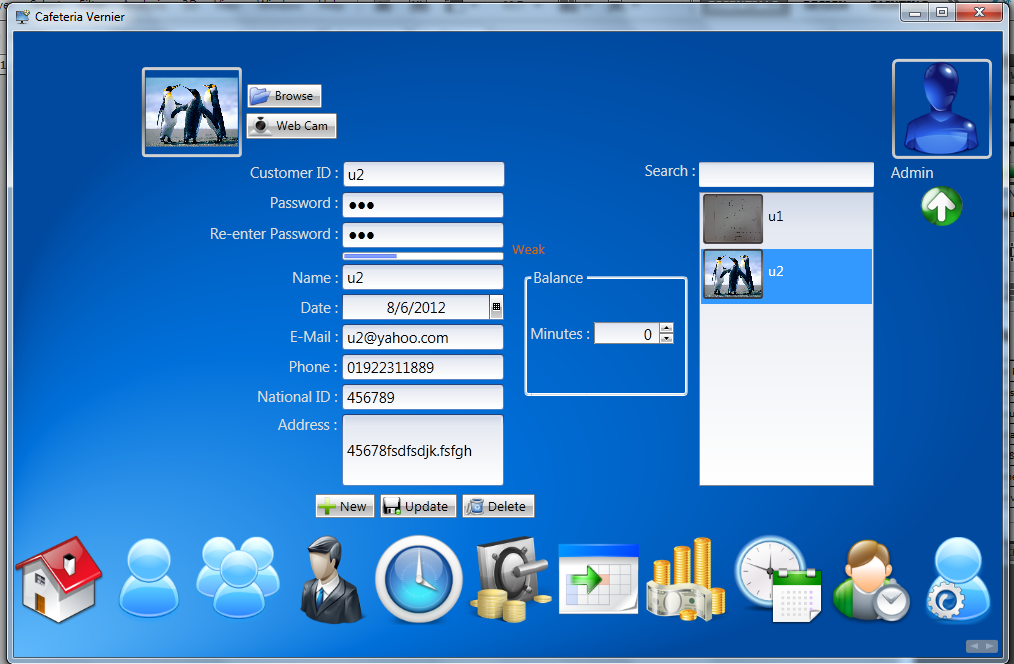


Important Screen short`s

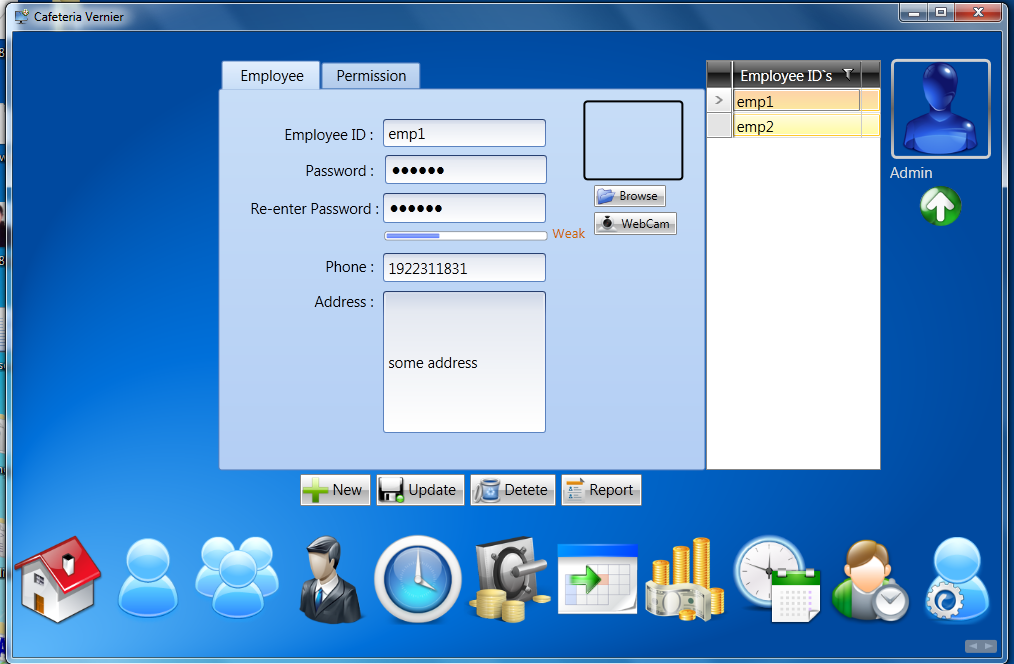
Counter`s Information Viewer :



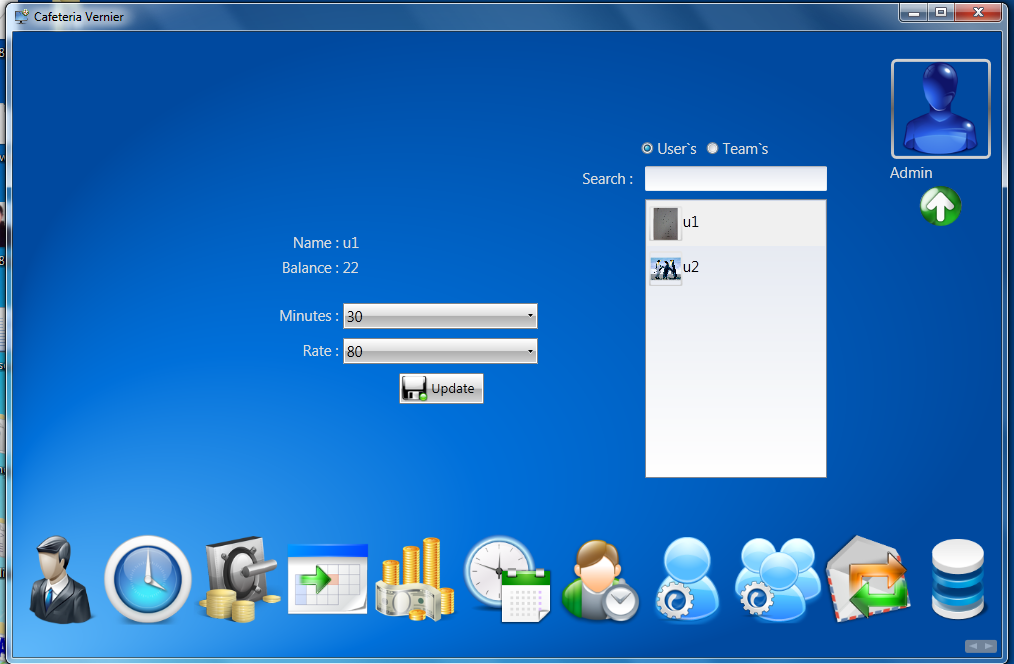
New Customer:



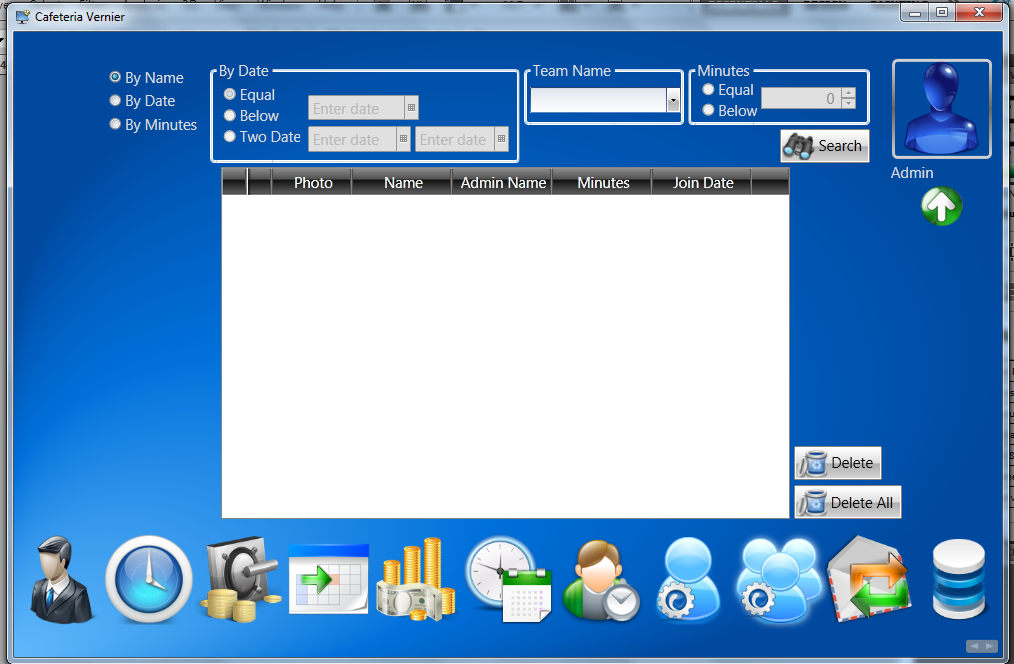
New Employee:



Account Recharges:



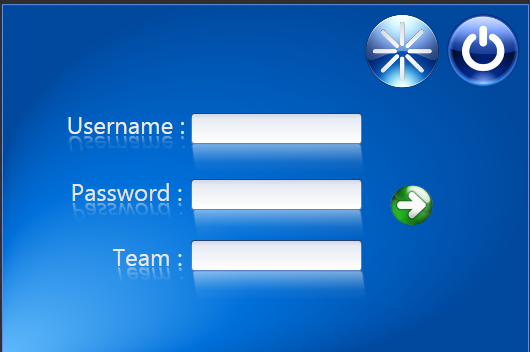
Team Account Maintenance:



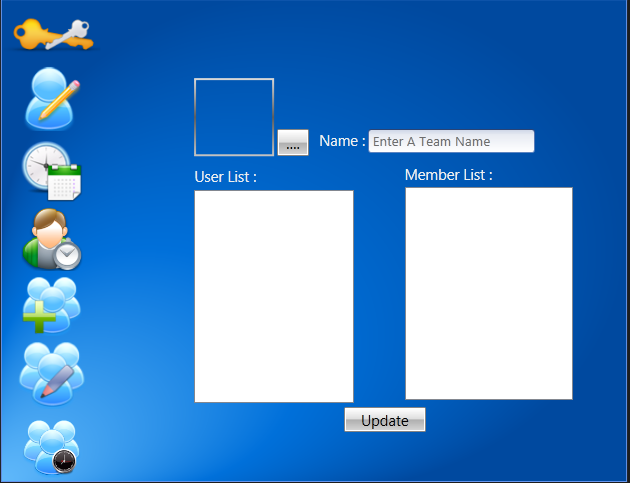
Rate Setup:



Customer Login:



Setting:



**Code**

**Server Side Code**

**Class: CounterSummary**

namespace Procesta.CvServer.Class.CounterInfo

{

public class CounterSummary : INotifyPropertyChanged

{

private string \_CounterName;

private string \_CounterIpAddress;

public string CounterName

{

get { return this.\_CounterName; }

set

{

this.\_CounterName = value;

this.OnPropertyChanged("CounterName");

}

}

public string CounterIpAddress

{

get { return this.\_CounterIpAddress; }

set

{

this.\_CounterIpAddress = value;

this.OnPropertyChanged("CounterIpAddress");

}

}

public event PropertyChangedEventHandler PropertyChanged;

private void OnPropertyChanged(string propertyName)

{

if (this.PropertyChanged != null)

{

// Raise the PropertyChanged event

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

}

**Class: AccountRechargeSearchs**

namespace Procesta.CvServer

{

class AccountRechargeSearchs

{

public AccountRechargeSearchs(ICollectionView filteredList, TextBox textEdit)

{

string filterText = string.Empty;

string properyText = string.Empty;

filteredList.Filter = delegate(object obj)

{

if (String.IsNullOrEmpty(filterText))

{

return true;

}

if (obj is ModelCustomer)

{

properyText = (obj as ModelCustomer).Name;

}

if (obj is ModelTeamInfo)

{

properyText = (obj as ModelTeamInfo).Name;

}

if (String.IsNullOrEmpty(properyText))

{

return true;

}

if (properyText.ToUpper().Contains(filterText.ToUpper()))

{

return true;

}

else

{

return false;

}

};

textEdit.TextChanged += delegate

{

filterText = textEdit.Text;

filteredList.Refresh();

};

}

}

}

Class: CommonInfoSearch

namespace Procesta.CvServer

{

class CommonInfoSearch

{

public CommonInfoSearch(ICollectionView filteredList, TextBox textEdit)

{

string filterText = string.Empty;

filteredList.Filter = delegate(object obj)

{

if (String.IsNullOrEmpty(filterText))

{

return true;

}

ModelCommonUse str = obj as ModelCommonUse;

if (str.UserName == null)

{

return true;

}

if (str.UserName.Trim().ToUpper().Contains(filterText.ToUpper()))

{

return true;

}

else

{

return false;

}

};

textEdit.TextChanged += delegate

{

filterText = textEdit.Text;

filteredList.Refresh();

};

}

}

}

**Class: CustomerInfoSearch**

namespace Procesta.CvServer

{

class CustomerInfoSearch

{

public CustomerInfoSearch(ICollectionView filteredList, TextBox textEdit)

{

string filterText = string.Empty;

filteredList.Filter = delegate(object obj)

{

if (String.IsNullOrEmpty(filterText))

{

return true;

}

ModelCustomer str = obj as ModelCustomer;

if (str.UserName==null)

{

return true;

}

if (str.UserName.ToUpper().Contains(filterText.ToUpper()))

{

return true;

}

else

{

return false;

}

};

textEdit.TextChanged += delegate

{

filterText = textEdit.Text;

filteredList.Refresh();

};

}

}

}

**Class: LogFileWriter**

namespace Procesta.CvServer

{

public class LogFileWriter

{

public static void ErrorToLog(string header, Exception occuredException)

{

try

{

LogFileWriter tempLogFileWriter = new LogFileWriter();

Task writelog = new Task(() => tempLogFileWriter.writeToLogFile(header, occuredException));

writelog.Start();

}

catch

{

throw;

}

}

private void writeToLogFile(string header, Exception occuredException)

{

try

{

string strLogMessage = string.Empty;

string folderPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "CVServer");

if (!Directory.Exists(folderPath))

{

Directory.CreateDirectory(folderPath);

}

string strLogFile = Path.Combine(folderPath, "CvServer.log");

StreamWriter swLog;

strLogMessage = string.Format("{0} : {1} => {2}", DateTime.Now, header, occuredException.Message);

if (!File.Exists(strLogFile))

{

swLog = new StreamWriter(strLogFile);

}

else

{

swLog = File.AppendText(strLogFile);

}

swLog.WriteLine(strLogMessage);

swLog.WriteLine();

swLog.Close();

}

catch (Exception)

{

throw;

}

}

}

}

**Class: NotificationFromClients**

namespace Procesta.CvServer.Class.Methods

{

public class NotificationFromClients : INotifyPropertyChanged

{

private ServerNotificationClient serverNotify = null;

private ObservableCollection<CounterInformation> \_CounterInformations;

public ObservableCollection<CounterInformation> CounterInformations

{

get

{

return this.\_CounterInformations;

}

set

{

this.\_CounterInformations = value;

if (this.PropertyChanged!=null)

{

this.PropertyChanged(this, new PropertyChangedEventArgs("CounterInformations"));

}

}

}

public NotificationFromClients()

{

this.CounterInformations = new ObservableCollection<CounterInformation>();

serverNotify = new ServerNotificationClient("NetNamedPipeBinding\_IServerNotification");

this.GetNotifactions();

}

private void GetNotifactions()

{

new Task(new Action(() =>

{

while (true)

{

try

{

this.CounterInformations = new ObservableCollection<CounterInformation>(this.serverNotify.GetCounterInformation());

foreach (CounterInformation counterInfo in this.CounterInformations)

{

var dd = DateTime.Now- counterInfo.sendingTime;

if ((DateTime.Now-counterInfo.sendingTime) > new TimeSpan(0, 0, 5))

{

this.serverNotify.RemoveCounterInformation(counterInfo);

}

}

System.Threading.Thread.Sleep(5000);

}

catch

{

break;

}

}

})).Start();

}

public event PropertyChangedEventHandler PropertyChanged;

}

}

**Class: ScreenShort**

namespace Procesta.CvServer.Class.Methods

{

public class ScreenShort

{

private static ThreadStart startScheenCapture;

private static Thread screenCaptureThread;

/// <summary>

/// Start Screenshot

/// </summary>

public static void start()

{

startScheenCapture = new ThreadStart(screenCapture);

screenCaptureThread = new Thread(startScheenCapture);

if (Properties.Settings.Default.IsCupture)

{

if (!screenCaptureThread.IsAlive)

{

screenCaptureThread.Start();

}

}

}

/// <summary>

/// Stop screenshot

/// </summary>

public static void Stop()

{

if (screenCaptureThread!=null && screenCaptureThread.IsAlive)

{

screenCaptureThread.Abort();

}

}

/// <summary>

/// Capture screen after a time

/// </summary>

private static void screenCapture()

{

try

{

while(true)

{

System.Drawing.Rectangle bounds = Screen.GetBounds(System.Drawing.Point.Empty);

using (Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height))

{

using (Graphics g = Graphics.FromImage(bitmap))

{

g.CopyFromScreen(new Point(bounds.Left, bounds.Top), Point.Empty, bounds.Size);

}

bitmap.Save(ImageNameMaker(), ImageFormat.Jpeg);

}

Thread.Sleep(Properties.Settings.Default.CuptureTime);

}

}

catch

{

return;

}

}

/// <summary>

/// Create a Image Name

/// </summary>

/// <returns></returns>

private static string ImageNameMaker()

{

string diris =System.IO.Path.Combine(Properties.Settings.Default.schreenCapturePath.Equals(string.Empty) ?

new MiraculousMethods().GetMyDocumentFolder("CV Screen Capture ") :

Properties.Settings.Default.schreenCapturePath,DateTime.Now.ToString("yyyyMMdd"));

if (!System.IO.Directory.Exists(diris))

{

System.IO.Directory.CreateDirectory(diris);

}

string fileNameIs = DateTime.Now.ToString("yyyyMMddHHmmss")+".jpg";

return System.IO.Path.Combine(diris, fileNameIs);

}

}

}

**Class: ModelCustomer**

namespace Procesta.CvServer

{

class ModelCustomer: INotifyPropertyChanged,IDataErrorInfo

{

public string UserName

{

get { return this.\_UserName; }

set

{

if (this.\_UserName != value)

{

this.\_UserName = value;

this.onPropertyChanged("UserName");

}

}

}

public string Password

{

get { return this.\_Password; }

set

{

if (this.\_Password != value)

{

this.\_Password = value;

this.onPropertyChanged("Password");

}

}

}

public string CheckPassword

{

get { return this.\_CheckPassword; }

set

{

if (this.\_CheckPassword != value)

{

this.\_CheckPassword = value;

this.onPropertyChanged("CheckPassword");

}

}

}

public string Name

{

get { return this.\_Name; }

set

{

if (this.\_Name != value)

{

this.\_Name = value;

this.onPropertyChanged("Name");

}

}

}

public string Phone

{

get { return this.\_Phone; }

set

{

if (this.\_Phone != value)

{

this.\_Phone = value;

this.onPropertyChanged("Phone");

}

}

}

public string Email

{

get { return this.\_Email; }

set

{

if (this.\_Email != value)

{

this.\_Email = value;

this.onPropertyChanged("Email");

}

}

}

public string NationalID

{

get { return this.\_NationalID; }

set

{

if (this.\_NationalID != value)

{

this.\_NationalID = value;

this.onPropertyChanged("NationalID");

}

}

}

public DateTime JoinDate

{

get { return this.\_JoinDate; }

set

{

if (this.\_JoinDate != value)

{

this.\_JoinDate = value;

this.onPropertyChanged("JoinDate");

}

}

}

public string Address

{

get { return this.\_Address; }

set

{

if (this.\_Address != value)

{

this.\_Address = value;

this.onPropertyChanged("Address");

}

}

}

public byte[] Image

{

get { return this.\_Image; }

set

{

if (this.\_Image != value)

{

this.\_Image = value;

this.onPropertyChanged("Image");

}

}

}

public int Minutes

{

get { return this.\_Minutes; }

set

{

if (this.\_Minutes != value)

{

this.\_Minutes = value;

this.onPropertyChanged("Minutes");

}

}

}

#region Private Variable

private string \_UserName;

private string \_Password;

private string \_CheckPassword;

private string \_Name;

private string \_Phone;

private string \_Email;

private string \_NationalID;

private DateTime \_JoinDate;

private string \_Address;

private byte[] \_Image;

private int \_Minutes;

#endregion

#region Property Change

public event PropertyChangedEventHandler PropertyChanged;

private void onPropertyChanged(string propertyName)

{

if (this.PropertyChanged!=null)

{

this.PropertyChanged(this,new PropertyChangedEventArgs(propertyName));

}

}

#endregion

#region Data Error

public string Error

{

get { return string.Empty; }

}

public string this[string columnName]

{

get

{

string errorMessagess = string.Empty;

switch (columnName)

{

case "UserName":

if (String.IsNullOrEmpty(this.UserName))

{

errorMessagess = String.Format(CvVariables.DEFULT\_ERROR\_FORMATE, "UserName");

}

break;

case "Password":

if (String.IsNullOrEmpty(this.Password))

{

errorMessagess = String.Format(CvVariables.DEFULT\_ERROR\_FORMATE, "Password");

}

else if( !(this.Password.Equals(this.CheckPassword)))

{

errorMessagess = "Password does not match.";

}

break;

case "Name":

if (String.IsNullOrEmpty(this.UserName))

{

errorMessagess = String.Format(CvVariables.DEFULT\_ERROR\_FORMATE, "Name");

}

break;

case "Phone":

if (String.IsNullOrEmpty(this.UserName))

{

errorMessagess = String.Format(CvVariables.DEFULT\_ERROR\_FORMATE, "Phone");

}

break;

default:

break;

}

return errorMessagess;

}

}

#endregion

}

}

Class: ModelEmployee

namespace Procesta.CvServer

{

public class ModelEmployee :INotifyPropertyChanged,IDataErrorInfo

{

public string Name

{

get { return this.\_Name; }

set

{

this.\_Name = value;

this.onPropertyChnage("Name");

}

}

public decimal? PhoneNmber

{

get { return this.\_PhoneNmber; }

set

{

this.\_PhoneNmber = value;

this.onPropertyChnage("PhoneNmber");

}

}

public string Password

{

get { return this.\_Password; }

set

{

this.\_Password = value;

this.onPropertyChnage("Password");

}

}

public string FirstPassword

{

get { return this.\_FirstPassword; }

set

{

if (this.\_FirstPassword != value)

{

this.\_FirstPassword = value;

this.onPropertyChnage("FirstPassword");

}

}

}

public string Address

{

get { return this.\_Address; }

set

{

this.\_Address = value;

this.onPropertyChnage("Address");

}

}

public byte[] UserImage

{

get { return this.\_UserImage; }

set

{

if (this.\_UserImage != value)

{

this.\_UserImage = value;

this.onPropertyChnage("UserImage");

}

}

}

public ObservableCollection<ModelEmployPermissions> Permissions

{

get { return this.\_Permissions; }

set

{

this.\_Permissions = value;

this.onPropertyChnage("Permissions");

}

}

#region Private Variable

private string \_Name;

private decimal? \_PhoneNmber;

private string \_Address;

private string \_Password;

private byte[] \_UserImage;

private string \_FirstPassword;

private ObservableCollection<ModelEmployPermissions> \_Permissions;

#endregion

#region Property Chnage

public event PropertyChangedEventHandler PropertyChanged;

private void onPropertyChnage(string propertyName)

{

if (this.PropertyChanged!=null)

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

#endregion

#region Error Info

public string Error

{

get { return string.Empty; }

}

public string this[string columnName]

{

get

{

string errorMessages=string.Empty;

switch (columnName)

{

case "Name":

if (String.IsNullOrEmpty(this.Name))

{

errorMessages = String.Format(CvVariables.DEFULT\_ERROR\_FORMATE,"Employee Name");

}

break;

case "PhoneNmber":

if (this.PhoneNmber==null)

{

errorMessages = String.Format(CvVariables.DEFULT\_ERROR\_FORMATE, "Employee Phone");

}

break;

case "Password":

if (String.IsNullOrEmpty(this.Password))

{

errorMessages = String.Format(CvVariables.DEFULT\_ERROR\_FORMATE, "Password");

}

else if(!(this.Password.Equals(this.FirstPassword)))

{

errorMessages = "Password does not match.";

}

break;

case "Address" :

if (String.IsNullOrEmpty(this.Address))

{

errorMessages = String.Format(CvVariables.DEFULT\_ERROR\_FORMATE, "Employee Address");

}

break;

default :

errorMessages = CvVariables.DEFAULT\_ERROR\_MESSAGES;

break;

}

return errorMessages;

}

}

#endregion

}

}

**Class: ModelTeamInfo**

namespace Procesta.CvServer

{

class ModelTeamInfo :INotifyPropertyChanged,IDataErrorInfo

{

public string Name

{

get { return this.\_Name; }

set

{

if (this.\_Name != value)

{

this.\_Name = value;

this.onPropertyChanged("Name");

}

}

}

public byte[] Image

{

get { return this.\_Image; }

set

{

if (this.\_Image != value)

{

this.\_Image = value;

this.onPropertyChanged("Image");

}

}

}

public string AdminName

{

get { return this.\_AdminName; }

set

{

if (this.\_AdminName != value)

{

this.\_AdminName = value;

this.onPropertyChanged("AdminName");

}

}

}

public DateTime JoinDate

{

get { return this.\_JoinDate; }

set

{

if (this.\_JoinDate != value)

{

this.\_JoinDate = value;

this.onPropertyChanged("JoinDate");

}

}

}

public ObservableCollection<ModelCommonUse> TeamMemberList

{

get { return this.\_TeamMemberList; }

set

{

if (this.\_TeamMemberList != value)

{

this.\_TeamMemberList = value;

this.onPropertyChanged("TeamMemberList");

}

}

}

public int Minutes

{

get { return this.\_Minutes; }

set

{

this.\_Minutes = value;

this.onPropertyChanged("Minutes");

}

}

#region Private Variables

private string \_Name;

private byte[] \_Image;

private string \_AdminName;

private DateTime \_JoinDate;

private ObservableCollection<ModelCommonUse> \_TeamMemberList;

private int \_Minutes;

#endregion

#region On Property Change

public event PropertyChangedEventHandler PropertyChanged;

private void onPropertyChanged(string PropertyName)

{

if (this.PropertyChanged!=null)

{

this.PropertyChanged(this, new PropertyChangedEventArgs(PropertyName));

}

}

#endregion

#region Data Error Info

public string Error

{

get { throw new NotImplementedException(); }

}

public string this[string columnName]

{

get

{

string errorMessagess = string.Empty;

switch (columnName)

{

case "Name":

if (String.IsNullOrEmpty(this.Name))

{

errorMessagess = String.Format(CvVariables.DEFULT\_ERROR\_FORMATE, "Team Name");

}

break;

case "AdminName":

if (String.IsNullOrEmpty(this.AdminName))

{

errorMessagess = String.Format(CvVariables.DEFULT\_ERROR\_FORMATE, "Admin name");

}

break;

default:

break;

}

return errorMessagess;

}

}

#endregion

}

}

**Class: CustomerUserName**

namespace Procesta.CvServer

{

public class CustomerUserName :ValidationRule

{

public override ValidationResult Validate(object value, System.Globalization.CultureInfo cultureInfo)

{

try

{

string customerUserName = (string)value;

if (String.IsNullOrWhiteSpace(customerUserName) || String.IsNullOrEmpty(customerUserName))

return new ValidationResult(false, "Customer username is no empty.");

using (Cafeteria\_Vernier\_dbEntities CVDatabase = new Cafeteria\_Vernier\_dbEntities())

{

if (CVDatabase.CustomerInformations.FirstOrDefault(CusUName => CusUName.UserID.Equals(customerUserName)) == null)

return new ValidationResult(true, null);

return new ValidationResult(false, "Customer username is not available.");

}

}

catch

{

return new ValidationResult(false, "Error occur during checking Customer username.");

}

}

}

}

Class: EmailAddressValidator

namespace Procesta.CvServer

{

public class EmailAddressValidator : ValidationRule

{

public override ValidationResult Validate(object value, System.Globalization.CultureInfo cultureInfo)

{

if (value!=null)

{

Regex emailRegex = new Regex(@"[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+(?:\.[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+)\*@(?:[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?");

if (emailRegex.Match(value.ToString()).Success)

{

return new ValidationResult(true, string.Empty);

}

else

{

return new ValidationResult(false, "Please enter a valid E-Mail address.");

}

}

else

{

return new ValidationResult(true, string.Empty);

}

}

}

}

**Class: EmployeeName**

namespace Procesta.CvServer

{

public class EmployeeName : ValidationRule

{

public override ValidationResult Validate(object value, System.Globalization.CultureInfo cultureInfo)

{

try

{

string employeeName = (string)value;

if (String.IsNullOrWhiteSpace(employeeName) || String.IsNullOrEmpty(employeeName))

return new ValidationResult(false, "Employee name is no empty.");

using (Cafeteria\_Vernier\_dbEntities CVDatabase = new Cafeteria\_Vernier\_dbEntities())

{

if (CVDatabase.Employees.FirstOrDefault(empName => empName.EmployeeID.Equals(employeeName)) == null)

return new ValidationResult(true, null);

return new ValidationResult(false, "Employee name is not available.");

}

}

catch

{

return new ValidationResult(false, "Error occur during checking employee name.");

}

}

}

}

**Class: PhoneNumber**

namespace Procesta.CvServer

{

public class PhoneNumber : ValidationRule

{

public override ValidationResult Validate(object value, System.Globalization.CultureInfo cultureInfo)

{

try

{

string phoneNumber = (string)value;

Regex emailRegex = new Regex(@"\b01[5-9|1][0-9]{8,8}\b");

if(emailRegex.IsMatch(phoneNumber))

return new ValidationResult(true, null);

return new ValidationResult(false, "Please enter a valid phone number.");

}

catch

{

return new ValidationResult(false, "Error occur during checking phone number.");

}

}

}

}

**Class: TeamName**

namespace Procesta.CvServer

{

public class TeamName :ValidationRule

{

public override ValidationResult Validate(object value, System.Globalization.CultureInfo cultureInfo)

{

try

{

string teamName = (string)value;

if (String.IsNullOrWhiteSpace(teamName) || String.IsNullOrEmpty(teamName))

return new ValidationResult(false, "Team name is no empty.");

using (Cafeteria\_Vernier\_dbEntities CVDatabase = new Cafeteria\_Vernier\_dbEntities())

{

if (CVDatabase.Teams.FirstOrDefault(tName => tName.Name.Equals(teamName)) == null)

return new ValidationResult(true, null);

return new ValidationResult(false, "Team name is not available.");

}

}

catch

{

return new ValidationResult(false, "Error occur during checking team name.");

}

}

}

}

**Class: ByteImageConverter**

namespace Procesta.CvServer

{

[ValueConversion(typeof(byte[]),typeof(BitmapImage))]

public class ByteImageConverter :IValueConverter

{

public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

byte[] binary = value as byte[];

if (binary != null)

{

using (var stream = new MemoryStream(binary.ToArray()))

{

BitmapImage bitmap = new BitmapImage();

bitmap.BeginInit();

bitmap.CacheOption = BitmapCacheOption.OnLoad;

bitmap.StreamSource = stream;

bitmap.EndInit();

bitmap.Freeze();

return bitmap;

}

}

return null;

}

public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

**Class: CounterStatusToImagePath**

namespace Procesta.CvServer

{

public class CounterStatusToImagePath : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

if (value !=null)

{

CounterStatus conStatus = (CounterStatus)value;

if (conStatus.Equals(CounterStatus.Free))

{

return "/Images/CounterFree.png";

}

else

{

return "/Images/CopunterBusy.png";

}

}

else

{

return "/Images/CounterFree.png";;

}

}

public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

**Class: Editpermission**

namespace Procesta.CvServer

{

public class Editpermission : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

if (value!=null)

{

return (value as ModelEmployee).Permissions.FirstOrDefault(x => x.Item.Equals(parameter.ToString())) != null ? true : false;

}

else

{

return false;

}

}

public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

**Class: MunitiesToColor**

namespace Procesta.CvServer.Class.ValueConverter

{

public class MunitiesToColor : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

if (value!=null)

{

Int32 munities =System.Convert.ToInt32(value);

if (munities <= 5)

{

return new SolidColorBrush(Color.FromRgb(0xFF, 0x00, 0x33));

}

else if (munities<=10)

{

return new SolidColorBrush(Colors.Yellow);

}

else

{

return new SolidColorBrush(Colors.Wheat);

}

}

else

{

return new SolidColorBrush(Colors.Wheat);

}

}

public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

**Class: PasswordsTolengthcs**

namespace Procesta.CvServer.Class.ValueConverter

{

[ValueConversion(typeof(string),typeof(double))]

public class PasswordsTolengthcs : IMultiValueConverter

{

public object Convert(object[] values, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

if (values[0]!=null && values[1]!=null && values[0].ToString()!=string.Empty && values[1].ToString()!=string.Empty)

{

if (values[0].ToString().Equals(values[1].ToString()))

{

if (values[1].ToString().Length <= 6)

{

return 1.0;

}

else if (values[1].ToString().Length <= 11)

{

return 2.0;

}

else

{

return 3.0;

}

}

else

{

return 0.0;

}

}

else

{

return 0.0;

}

}

public object[] ConvertBack(object value, Type[] targetTypes, object parameter, System.Globalization.CultureInfo culture)

{

object[] obj = new object[1];

return obj;

}

}

}

**Class: PasswordsToStrength**

namespace Procesta.CvServer.Class.ValueConverter

{

public class PasswordsToStrength : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

if (value!=null && ((double)value)> 0.0)

{

if (((double)value).Equals(1.0))

{

return "Weak";

}

else if (((double)value).Equals(2.0))

{

return "Medium";

}

else

{

return "Strong";

}

}

else

{

return string.Empty;

}

}

public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

**Class: RadioToString**

namespace Procesta.CvServer

{

public class RadioToString : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

if (value != null && parameter != null)

{

string checkValue = value.ToString();

string targetValue = parameter.ToString();

return checkValue.Equals(targetValue, StringComparison.InvariantCultureIgnoreCase);

}

else

{

return false;

}

}

public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

if (value == null || parameter == null)

return Binding.DoNothing;

bool useValue = (bool)value;

if (useValue)

{

return parameter;

}

return Binding.DoNothing;

}

}

}

Class: RechargerDatacontextChanger

namespace Procesta.CvServer

{

/// <summary>

/// value[0] : radioButtonUser

/// value[1] : radioButtonTeam

/// value[2] : AccountRecUser.SelectedItem

/// value[3] : AccountRecTeam.SelectedItem

/// </summary>

public class RechargerDatacontextChanger : IMultiValueConverter

{

public object Convert(object[] values, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

if (values!=null)

{

try

{

if ((bool)values[0])

{

return values[2];

}

else if((bool)values[1])

{

return values[3];

}

else

{

return Binding.DoNothing;

}

}

catch

{

return Binding.DoNothing;

}

}

else

{

return Binding.DoNothing;

}

}

public object[] ConvertBack(object value, Type[] targetTypes, object parameter, System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

**Class: ValuePacker**

namespace Procesta.CvServer

{

public class ValuePacker : IMultiValueConverter

{

public object Convert(object[] values, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

ArrayList packingValues = new ArrayList();

foreach (object value in values)

{

packingValues.Add(value);

}

return packingValues;

}

public object[] ConvertBack(object value, Type[] targetTypes, object parameter, System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

**Class: SnapShotTimeCollections**

namespace Procesta.CvServer

{

public class SnapShotTimeCollections : ObservableCollection<ModelSnapShotTime>

{

public SnapShotTimeCollections()

{

this.Add(new ModelSnapShotTime { Name = "30 Seconds", shotTime = new TimeSpan(0, 0,30) });

this.Add(new ModelSnapShotTime { Name = "1 Minutes", shotTime = new TimeSpan(0,1,0) });

this.Add(new ModelSnapShotTime { Name = "5 Minutes", shotTime = new TimeSpan(0, 5, 0) });

this.Add(new ModelSnapShotTime { Name = "10 Minutes", shotTime = new TimeSpan(0, 10, 0) });

this.Add(new ModelSnapShotTime { Name = "20 Minutes", shotTime = new TimeSpan(0, 20, 0) });

this.Add(new ModelSnapShotTime { Name = "30 Minutes", shotTime = new TimeSpan(0, 30, 0) });

this.Add(new ModelSnapShotTime { Name = "1 Hour", shotTime = new TimeSpan(1, 0, 0) });

this.Add(new ModelSnapShotTime { Name = "2 Hours", shotTime = new TimeSpan(2, 0, 0) });

}

}

}

**Class: ViewEmployPermissions**

namespace Procesta.CvServer

{

class ViewEmployPermissions : ObservableCollection<ModelEmployPermissions>

{

public ViewEmployPermissions()

{

this.Add(new ModelEmployPermissions { Item = "NewCustomer", Setting = null, Permission = true, Priority = 2, SupperTip = "Create Customer Account", ScreenTip = "Add a new customer", KeboardShortcut = "Ctrl + C", ImagePath = "/ButtonImages/Customer.png" });

this.Add(new ModelEmployPermissions { Item = "EditCustomer", Setting = null, Permission = false, Priority = 0, SupperTip = "Edit Customer Profile", ScreenTip = null, KeboardShortcut = null, ImagePath = null });

this.Add(new ModelEmployPermissions { Item = "AccountRecharge", Setting = null, Permission = true, Priority = 5, SupperTip = "Account Recharge", ScreenTip = "Recharge customers account.", KeboardShortcut = "Ctrl + R", ImagePath = "/ButtonImages/Recharge.png" });

this.Add(new ModelEmployPermissions { Item = "EditRecharge", Setting = null, Permission = false, Priority = 0, SupperTip = "Edit Account Balance", ScreenTip = null, KeboardShortcut = null, ImagePath = null });

this.Add(new ModelEmployPermissions { Item = "RechargeHistory", Setting = null, Permission = true, Priority = 9, SupperTip = "View Recharge History", ScreenTip = "View customers account recharge history", KeboardShortcut = "Ctrl + E", ImagePath = "/ButtonImages/RechargeHistory.png" });

this.Add(new ModelEmployPermissions { Item = "CustomerLoginHistory", Setting = null, Permission = true, Priority = 10, SupperTip = "View Login History", ScreenTip = "View customers login history", KeboardShortcut = "Ctrl + L", ImagePath = "/ButtonImages/UserLoginHistory.png" });

this.Add(new ModelEmployPermissions { Item = "Cash", Setting = null, Permission = true, Priority = 6, SupperTip = "Cash", ScreenTip = "View Cash", KeboardShortcut = "Ctrl + A", ImagePath = "/ButtonImages/CashView.png" });

this.Add(new ModelEmployPermissions { Item = "EditCash", Setting = null, Permission = false, Priority = 0, ScreenTip = "Edit Cash", SupperTip = "Edit Cash Info", KeboardShortcut = null, ImagePath = null });

this.Add(new ModelEmployPermissions { Item = "Summary", Setting = null, Permission = true, Priority = 7, SupperTip = "View Business Summary", ScreenTip = "View business summary by date", KeboardShortcut = "Ctrl + S", ImagePath = "/ButtonImages/TodaySummary.png" });

this.Add(new ModelEmployPermissions { Item = "Setting", Setting = "RateSetting", Permission = false, Priority = 19, SupperTip = "Rate Setup", ScreenTip = "Setup your rate", KeboardShortcut = "Ctrl + U", ImagePath = "/ButtonImages/MoneySetting.png" });

this.Add(new ModelEmployPermissions { Item = "Setting", Setting = null, Permission = true, Priority = 17, SupperTip = "Setting", ScreenTip = "Application setting", KeboardShortcut = "Ctrl + I", ImagePath = "/ButtonImages/settings.png" });

this.Add(new ModelEmployPermissions { Item = "CustomerMaintenance", Setting = null, Permission = false, Priority = 11, SupperTip = "Customer Account Maintenance", ScreenTip = "Maintenance your customer`s account`s", KeboardShortcut = "Ctrl + O", ImagePath = "/ButtonImages/userMaintenance.png" });

this.Add(new ModelEmployPermissions { Item = "TeamMaintenance", Setting = null, Permission = false, Priority = 12, SupperTip = "Team Account Maintenance", ScreenTip = "Maintenance team`s account`s", KeboardShortcut = "Ctrl + N", ImagePath = "/ButtonImages/TeamMaintenance.png" });

this.Add(new ModelEmployPermissions { Item = "NewTeam", Setting = null, Permission = true, Priority = 3, SupperTip = "Create Team Account", ScreenTip = "Add New Team", KeboardShortcut = "Ctrl + M", ImagePath = "/ButtonImages/Team.png" });

this.Add(new ModelEmployPermissions { Item = "Setting", Setting = "ChangePassword", Permission = true, Priority = 18, SupperTip = "Change Password", ScreenTip = "Change your Password", KeboardShortcut = "Ctrl + P", ImagePath = "/ButtonImages/ChangePassword.png" });

this.Add(new ModelEmployPermissions { Item = "Setting", Setting = "ScreenCapture", Permission = false, Priority = 21, SupperTip = "Screen Snapshot", ScreenTip = "Capture host screen.", KeboardShortcut = "Ctrl + F", ImagePath = "/ButtonImages/ScreenCaptureSetting.png" });

this.Add(new ModelEmployPermissions { Item = "SendEmail", Setting = null, Permission = true, Priority = 13, SupperTip = "Send E-Mail", ScreenTip = "Send E-Mail to your client`s", KeboardShortcut = "Ctrl + J", ImagePath = "/ButtonImages/SenEmail.png" });

this.Add(new ModelEmployPermissions { Item = "Setting", Setting = "EMailSetting", Permission = false, Priority = 20, SupperTip = "E-Mail Setup", ScreenTip = "Change your E-Mail.", KeboardShortcut = "Ctrl + K", ImagePath = "/ButtonImages/EmailSetting.png" });// Image path need

this.Add(new ModelEmployPermissions { Item = "Setting", Setting = "GeneralSetting", Permission = true, Priority = 17, SupperTip = "General Setting", ScreenTip = "Setup application general setting", KeboardShortcut = "Ctrl + G", ImagePath = "/ButtonImages/General.png" });

this.Add(new ModelEmployPermissions { Item = "CountersInformation", Setting = null, Permission = true, Priority = 1, SupperTip = "View Counter`s Information", ScreenTip = "Counter`s Information", KeboardShortcut = "Ctrl + H", ImagePath = "/ButtonImages/Home.png" });

this.Add(new ModelEmployPermissions { Item = "Setting", Setting = "CounterSetting", Permission = false, Priority = 21, SupperTip = "Counter Setting", ScreenTip = "Setup tour counters", KeboardShortcut = "Ctrl + T", ImagePath = "/ButtonImages/counterSetting.png" });

this.Add(new ModelEmployPermissions { Item = "CashHistory", Setting = null, Permission = true, Priority = 8, SupperTip = "Cash History", ScreenTip = "See cash history", KeboardShortcut = "Ctrl + Y", ImagePath = "/ButtonImages/CashHHistory.png" });

this.Add(new ModelEmployPermissions { Item = "NewEmployee", Setting = null, Permission = false, Priority = 4, SupperTip = "Create Employee Account", ScreenTip = "Add new employee", KeboardShortcut = "Ctrl + Q", ImagePath = "/ButtonImages/Employee.png" });

this.Add(new ModelEmployPermissions { Item = "EmployeeEdit", Setting = null, Permission = false, Priority = 0, SupperTip = "Edit Employee Profile", ScreenTip = null, KeboardShortcut = null, ImagePath = null });

this.Add(new ModelEmployPermissions { Item = "Database", Setting = null, Permission = false, Priority = 15, SupperTip = "Database Operation", ScreenTip = "Backup and Restore your database.", KeboardShortcut = "Ctrl + D", ImagePath = "/ButtonImages/database.png" });

this.Add(new ModelEmployPermissions { Item = "CustomerStatusReset", Setting = null, Permission = true, Priority = 16, SupperTip = "Login Status Reset", ScreenTip = "Reset your customer login status", KeboardShortcut = "Ctrl + S", ImagePath = "/ButtonImages/UserReset.png" });

}

}

}

**Class: CvVariables**

namespace Procesta.CvServer

{

public class CvVariables

{

public const string Catalog = "Cafeteria\_Vernier\_db";

public const string SOFTWARE\_NAME = "Cafeteria Vernier";

public const string MUNITIES\_FILE = "BillConfig.xml";

public const string COUNTER\_CONFIG = "countConfig.xml";

public static string SQL\_SERVER\_NAME = @".\SQLExpress";

public const string DEFAULT\_ERROR\_MESSAGES = "Unwanted error.";

public const string DEFULT\_ERROR\_FORMATE = "{0} is required.";

public static String[,] ERROR\_MESSAGES = new String[,]

{

{"Cafeteria Vernier","Invalid Command","Login Failed\nUsername or password may wrong.","Error ! While building menu.","Are you sure you want to Exit ?","Error ! While converting","Logout ! are you sure ?","Password does not match","Weak","Medium","Strong"},

// 00 01 02 03 04 05 06 07 08 09 010

{"Image size must be less than 800 KB","Username is not available !","Registration successfully done ! ","User not found","Update successfully done !","Error ! \n Please try again.","Deleted successfully !","File not found !","Are you sure you want to Update ?","Successfully added !","Please select an user"},

// 10 11 12 13 14 15 16 17 18 19 110

{"E-Mail`s has been sent !","Configuration Finished at","Configuration Unfinished at","successfully done","Access denied","Database backup finished.","Database restore finished.","Are you sure you want to overwrite database ?","Member already exits.","Invalid phone number.","Invalid E-mail address."},

// 20 21 22 23 24 25 26 27 28 29 210

{"Team is not available !","Invalid IP address","","","","","","","","",""}

// 30 31

};

public static string[,] ERROR\_MESSAGESS = new string[,]

{

{"Invalid Command","Access permission required.","Update successfully done.","Registration successfully done.","Please select an user.","Deleted successfully !","Are you sure are you want to delete ?","Database successfully backup.","Database successfully restore.","Password change successfully.","Please select another place."}

// 00 01 02 03 04 05 06 07 08 09 010

};

public static String[] MENU\_PERMISSION = new String[] { "New Customer", "Customer Edit", "Recharge Account", "Recharge Edit", "Recharge History", "User Login History", "Cash", "Cash Edit", "Summary", "Setting", "MoneySetting", "User Maintenance", "Team Maintenance", "New Team", "Change Password", "Screen Capture", "Send Email", "Email Setting", "General Setting", "Home", "Counter Setting", "Counter Config", "New Employ", "Employ Edit", "Database Backup", "UserStatusReset" };

// 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

}

}

**Class: InstallWindow**

namespace Procesta.CvServer

{

/// <summary>

/// Interaction logic for InstallWindow.xaml

/// </summary>

public partial class InstallWindow : Window

{

public InstallWindow()

{

InitializeComponent();

}

/// <summary>

/// Save Button Click

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void IpaddressButtonSaveClick(object sender, System.Windows.RoutedEventArgs e)

{

this.IpaddressButtonSave.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

bool isDatabaseFound = false;

try

{

if (this.IpAddressComBoBox.Text != string.Empty && !String.IsNullOrWhiteSpace(this.IpAddressComBoBox.Text))

{

Properties.Settings.Default.IsConfigered = true;

Server databaseServer = new Server(new ServerConnection(CvVariables.SQL\_SERVER\_NAME));

foreach (Database serverDatabase in databaseServer.Databases)

{

if (serverDatabase.Name.Equals(CvVariables.Catalog))

{

isDatabaseFound = true;

}

}

if (isDatabaseFound)

{

Mouse.OverrideCursor = null;

MessageBoxResult messBoxResult = DXMessageBox.Show(CvVariables.ERROR\_MESSAGES[2, 7], CvVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.YesNo, MessageBoxImage.Question);

Mouse.OverrideCursor = Cursors.Wait;

if (messBoxResult.Equals(MessageBoxResult.Yes))

{

this.RestoreDatabase();

}

}

else

{

this.RestoreDatabase();

}

Properties.Settings.Default.Save();

System.Windows.Forms.Application.Restart();

System.Windows.Application.Current.Shutdown();

this.Close();

}

else

{

DXMessageBox.Show(CvVariables.ERROR\_MESSAGES[0, 1],CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Exclamation);

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CvVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.IpaddressButtonSave.IsEnabled = true;

}

}

/// <summary>

/// Window Load Event

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void IpAddressWindowsLoaded(object sender, System.Windows.RoutedEventArgs e)

{

IPAddress[] localPs = Dns.GetHostAddresses(Dns.GetHostName());

foreach (IPAddress WorkingIp in localPs)

{

this.IpAddressComBoBox.Items.Add(WorkingIp);

}

}

/// <summary>

/// Restore Database at First time Setup

/// </summary>

private void RestoreDatabase()

{

try

{

using (SqlConnection restoreConnection = new SqlConnection(@"Data Source=.\SQLEXPRESS;Initial Catalog=master;Integrated Security=True"))

{

restoreConnection.Open();

SqlCommand UseMasterCommand = new SqlCommand("USE master", restoreConnection);

UseMasterCommand.ExecuteNonQuery();

string Restore = string.Format(@"RESTORE DATABASE [{0}] FROM DISK = N'{1}' WITH FILE = 1, NOUNLOAD, STATS = 10,MOVE '{0}' TO " + @"'{2}\{0}.mdf',MOVE '{0}\_log' TO '{2}\{0}\_log.ldf'", CvVariables.Catalog, System.IO.Path.Combine(System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetEntryAssembly().Location), "cvdb.bak"), Properties.Settings.Default.SqlDataFolder);

SqlCommand RestoreCmd = new SqlCommand(Restore, restoreConnection);

RestoreCmd.ExecuteNonQuery();

string Alter2 =string.Format(@"ALTER DATABASE [{0}] SET Multi\_User",CvVariables.Catalog);

SqlCommand Alter2Cmd = new SqlCommand(Alter2, restoreConnection);

Alter2Cmd.ExecuteNonQuery();

}

}

catch (Exception error)

{

throw error;

}

}

/// <summary>

/// Brows Button Click

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void configButtonBrowsClick(object sender, System.Windows.RoutedEventArgs e)

{

this.configButtonBrows.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait; ;

try

{

System.Windows.Forms.FolderBrowserDialog backupPathDialog = new System.Windows.Forms.FolderBrowserDialog();

backupPathDialog.Description = "Select SQl Server DATA Folder";

backupPathDialog.ShowNewFolderButton = false;

if (backupPathDialog.ShowDialog().Equals(System.Windows.Forms.DialogResult.OK))

{

this.ConfigDataFolderPath.Text = backupPathDialog.SelectedPath;

Properties.Settings.Default.SqlDataFolder=backupPathDialog.SelectedPath;

Properties.Settings.Default.Save();

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CvVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.configButtonBrows.IsEnabled = true;

}

}

}

}

Class: MainWindow

namespace Procesta.CvServer

{

/// <summary>

/// Interaction logic for Window1.xaml

/// </summary>`

public partial class MainWindow : Window, INotifyPropertyChanged

{

#region Private Variables

private BackgroundWorker sendMailWorker = new BackgroundWorker();

private BackgroundWorker dbBackupWorker = new BackgroundWorker();

private BackgroundWorker dbRestoreWorker = new BackgroundWorker();

private ModelEmployee \_LoginEmployee;

private ObservableCollection<ModelBillConfig> \_BillConfigInfo;

private string \_Option = "ByDate";

private string \_SubOption = "Equal";

#endregion

#region Propertys

public ModelEmployee LoginEmployee

{

get { return this.\_LoginEmployee; }

set

{

this.\_LoginEmployee = value;

this.OnPropertyChanged("LoginEmployee");

}

}

public ObservableCollection<ModelBillConfig> BillConfigInfo

{

get { return this.\_BillConfigInfo; }

set

{

if (this.\_BillConfigInfo != value)

{

this.\_BillConfigInfo = value;

this.OnPropertyChanged("BillConfigInfo");

}

}

}

public string Option

{

get { return this.\_Option; }

set

{

if (this.\_Option != value)

{

this.\_Option = value;

this.OnPropertyChanged("Option");

}

}

}

public string SubOption

{

get { return this.\_SubOption; }

set

{

if (this.\_SubOption != value)

{

this.\_SubOption = value;

this.OnPropertyChanged("SubOption");

}

}

}

#endregion

public MainWindow()

{

InitializeComponent();

this.DataContext = this;

sendMailWorker.WorkerReportsProgress = true;

sendMailWorker.WorkerSupportsCancellation = true;

sendMailWorker.DoWork += new DoWorkEventHandler(sendMailWorker\_DoWork);

sendMailWorker.ProgressChanged += new ProgressChangedEventHandler(sendMailWorker\_ProgressChanged);

sendMailWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(sendMailWorker\_RunWorkerCompleted);

dbBackupWorker.WorkerReportsProgress = true;

dbBackupWorker.DoWork += new DoWorkEventHandler(dbBackupWorker\_DoWork);

dbBackupWorker.ProgressChanged += new ProgressChangedEventHandler(dbBackupWorker\_ProgressChanged);

dbBackupWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(dbBackupWorker\_RunWorkerCompleted);

dbRestoreWorker.WorkerReportsProgress = true;

dbRestoreWorker.DoWork += new DoWorkEventHandler(dbRestoreWorker\_DoWork);

dbRestoreWorker.ProgressChanged += new ProgressChangedEventHandler(dbRestoreWorker\_ProgressChanged);

dbRestoreWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(dbRestoreWorker\_RunWorkerCompleted);

}

private void Window\_Load(object sender, System.Windows.RoutedEventArgs e)

{

this.LoginBusyIndicator.Visibility = Visibility.Visible;

}

#region Counter Information

public ICommand ClientShoutdownCommand

{

get { return new ReplayCommand(new Action<object>(this.clientShoutdown\_Click)); }

}

public ICommand ClientRestartCommand

{

get { return new ReplayCommand(new Action<object>(this.clientRestart\_Click)); }

}

public ICommand ClientLogoffCommand

{

get { return new ReplayCommand(new Action<object>(this.clientLogoff\_Click)); }

}

private void clientShoutdown\_Click(object obj)

{

ClientNotificationClient clientNotify = new ClientNotificationClient("NetNamedPipeBinding\_IClientNotification");

clientNotify.setCommand(new CommandData { Command = Commands.Shutdown, CounterNumber = obj.ToString() });

clientNotify.Close();

}

private void clientRestart\_Click(object obj)

{

ClientNotificationClient clientNotify = new ClientNotificationClient("NetNamedPipeBinding\_IClientNotification");

clientNotify.setCommand(new CommandData { Command = Commands.Restart, CounterNumber = obj.ToString() });

clientNotify.Close();

}

private void clientLogoff\_Click(object obj)

{

ClientNotificationClient clientNotify = new ClientNotificationClient("NetNamedPipeBinding\_IClientNotification");

clientNotify.setCommand(new CommandData { Command = Commands.AccountLogout, CounterNumber = obj.ToString() });

clientNotify.Close();

}

private void tileView1\_TileStateChanged(object sender, RadRoutedEventArgs e)

{

RadTileViewItem item = e.OriginalSource as RadTileViewItem;

if (item != null)

{

RadFluidContentControl fluid = item.ChildrenOfType<RadFluidContentControl>().FirstOrDefault();

if (fluid != null)

{

switch (item.TileState)

{

case TileViewItemState.Maximized:

fluid.State = FluidContentControlState.Large;

break;

case TileViewItemState.Minimized:

fluid.State = FluidContentControlState.Normal;

break;

case TileViewItemState.Restored:

fluid.State = FluidContentControlState.Normal;

break;

default:

break;

}

}

}

}

#endregion

#region Menu Button

public ICommand MainMenuCommand

{

get { return new ReplayCommand(new Action<object>(this.mainMenuClick)); }

}

private void mainMenuClick(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (this.LoginEmployee==null || this.LoginEmployee.Permissions.AsParallel().FirstOrDefault(x => x.Item.Equals(obj.ToString())) == null)

{

return;

}

this.hidePanels();

switch (obj.ToString())

{

case "NewEmployee":

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities CVDatabase = new Cafeteria\_Vernier\_dbEntities())

{

ViewEmployPermissions defultParmissions = new ViewEmployPermissions();

ObservableCollection<ModelEmployee> employeeinfoList = new ObservableCollection<ModelEmployee>(from employeeInfo in CVDatabase.Employees.ToList().Where(x => x.EmployeeID.Trim() != "Admin")

select new ModelEmployee

{

UserImage = employeeInfo.UserImage,

Address = employeeInfo.Address,

Name = employeeInfo.EmployeeID,

Password = employeeInfo.Password,

FirstPassword = employeeInfo.Password,

PhoneNmber = employeeInfo.Phone,

Permissions = new ObservableCollection<ModelEmployPermissions>

(

(from permissionDb in defultParmissions select new ModelEmployPermissions { SupperTip = permissionDb.SupperTip, Priority = permissionDb.Priority, Setting = permissionDb.Setting, Permission = employeeInfo.EmployeePermissions.FirstOrDefault(x => x.Privilege.Trim().Equals(permissionDb.Item, StringComparison.InvariantCultureIgnoreCase) && permissionDb.Setting == x.SettingPrivilage) != null ? true : false }).OrderBy(x => x.Priority)

)

});

this.Dispatcher.BeginInvoke(

new Action(() =>

{

this.EmployeeGridView.ItemsSource = employeeinfoList;

this.selectGridViewFirstItem(this.EmployeeGridView);

}),DispatcherPriority.DataBind);

}

})).Start();

this.PanelNewEmploy.Visibility = Visibility.Visible;

break;

case "CountersInformation":

this.PanelCounterView.Visibility = Visibility.Visible;

break;

case "NewCustomer":

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities CVDatabase = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<ModelCustomer> customerCollection = new ObservableCollection<ModelCustomer>(from customerInfo in CVDatabase.CustomerInformations

select new ModelCustomer

{

UserName = customerInfo.UserID,

Address = customerInfo.Address,

Email = customerInfo.Email,

Image = customerInfo.Logo,

JoinDate = customerInfo.JoinDate,

NationalID = customerInfo.NationalID,

Name = customerInfo.Name,

Phone = customerInfo.Phone,

Minutes = customerInfo.CustomerAccount.Minutes,

CheckPassword = customerInfo.CustomerAccount.Password,

Password = customerInfo.CustomerAccount.Password

});

this.Dispatcher.BeginInvoke(

new Action(() =>

{

ICollectionView ProductInfoView = CollectionViewSource.GetDefaultView(customerCollection);

this.CustomerList.ItemsSource = customerCollection;

this.selectListBoxFirstItem(this.CustomerList);

new CustomerInfoSearch(ProductInfoView, this.CustomerTxtSearch);

}),DispatcherPriority.DataBind);

}

})).Start();

this.PanelNewCustomer.Visibility = Visibility.Visible;

break;

case "NewTeam":

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

var teamsQuery = new ObservableCollection<ModelTeamInfo>(from teamInfo in cvDatabase.Teams.ToList()

select new ModelTeamInfo

{

AdminName = teamInfo.AdminName,

Image = teamInfo.Logo,

JoinDate = teamInfo.JoinDate,

Minutes = teamInfo.TeamAccount.Minutes,

Name = teamInfo.Name,

TeamMemberList = new ObservableCollection<ModelCommonUse>(from member in teamInfo.TeamMembers

where member.Name!=null

select new ModelCommonUse

{

UserName=member.UserID,

Image=member.CustomerInformation.Logo

})

});

this.Dispatcher.BeginInvoke(new Action(() =>

{

this.TeamGridView.ItemsSource = teamsQuery;

this.selectGridViewFirstItem(this.TeamGridView);

}), DispatcherPriority.DataBind);

}

})).Start();

new Task(

new Action(() =>

{

ObservableCollection<ModelCommonUse> customerShotInfo = this.customerInfo();

this.Dispatcher.BeginInvoke(

new Action(() =>

{

ICollectionView userInfoView = CollectionViewSource.GetDefaultView(customerShotInfo);

new CommonInfoSearch(userInfoView, this.TeamUserSearch);

this.TeamExistUserList.ItemsSource = customerShotInfo;

this.TeamAdminName.ItemsSource = customerShotInfo;

}),DispatcherPriority.DataBind);

})).Start();

this.PanelNewTeam.Visibility = Visibility.Visible;

break;

case "AccountRecharge":

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<ModelCustomer> userQuery = new ObservableCollection<ModelCustomer>(from userInfo in cvDatabase.CustomerInformations

select new ModelCustomer

{

Image = userInfo.Logo,

Name = userInfo.UserID,

Minutes=userInfo.CustomerAccount.Minutes

});

this.Dispatcher.BeginInvoke(

new Action(() =>

{

ICollectionView userInfoView = CollectionViewSource.GetDefaultView(userQuery);

new AccountRechargeSearchs(userInfoView, this.AccountRecUserSearch);

this.AccountRecUser.ItemsSource = userQuery;

this.selectListBoxFirstItem(this.AccountRecUser);

}));

}

})).Start();

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<ModelTeamInfo> teamQuery = new ObservableCollection<ModelTeamInfo>(from teamInfo in cvDatabase.Teams

select new ModelTeamInfo

{

Image = teamInfo.Logo,

Name = teamInfo.Name,

Minutes=teamInfo.TeamAccount.Minutes

});

this.Dispatcher.BeginInvoke(

new Action(() =>

{

ICollectionView teamInfoView = CollectionViewSource.GetDefaultView(teamQuery);

new AccountRechargeSearchs(teamInfoView, this.AccountRecTeamSearch);

this.AccountRecTeam.ItemsSource = teamQuery;

this.selectListBoxFirstItem(this.AccountRecTeam);

}));

}

})).Start();

this.PanelAccountRecharge.Visibility = Visibility.Visible;

break;

case "Cash":

this.PanelCahView.Visibility = Visibility.Visible;

break;

case "CashHistory":

this.PanelCashHistory.Visibility = Visibility.Visible;

break;

case "Summary":

this.PanelSummary.Visibility = Visibility.Visible;

break;

case "RechargeHistory":

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities cvDatabse = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<ModelCommonUse> cutomerShortInfo = this.customerInfo();

this.Dispatcher.BeginInvoke(

new Action(() =>

{

this.ResHisCustomerComboBox.ItemsSource = cutomerShortInfo;

}));

}

})).Start();

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities cvDatabse = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<ModelCommonUse> teamShortInfo = new ObservableCollection<ModelCommonUse>(cvDatabse.Teams.Select(x =>

new ModelCommonUse

{

UserName = x.Name,

Image = x.Logo

}));

this.Dispatcher.BeginInvoke(

new Action(() =>

{

this.ResHisTeamComboBox.ItemsSource = teamShortInfo;

}));

}

})).Start();

this.PanelRechargeHistoryView.Visibility = Visibility.Visible;

break;

case "CustomerLoginHistory":

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities cvDatabse = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<ModelCommonUse> customerShortInfo = this.customerInfo();

this.Dispatcher.BeginInvoke(

new Action(() =>

{

this.LogHisComboBox.ItemsSource = customerShortInfo;

}));

}

})).Start();

this.PanelLoginHistory.Visibility = Visibility.Visible;

break;

case "CustomerMaintenance":

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities cvDatabse = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<ModelCommonUse> cusromerShortInfo = this.customerInfo();

this.Dispatcher.BeginInvoke(

new Action(() =>

{

this.UserMaintenanceCutomer.ItemsSource = cusromerShortInfo;

}));

}

})).Start();

this.Option = "ByName";

this.SubOption = "ByDate";

this.PanelUserMaintenance.Visibility = Visibility.Visible;

break;

case "TeamMaintenance":

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities cvDatabse = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<ModelCommonUse> teamShortInfo = new ObservableCollection<ModelCommonUse>(cvDatabse.Teams.Select(x =>

new ModelCommonUse

{

UserName = x.Name,

Image = x.Logo

}));

this.Dispatcher.BeginInvoke(

new Action(() =>

{

this.temMainTeamName.ItemsSource = teamShortInfo;

}));

}

})).Start();

this.Option = "ByName";

this.SubOption = "ByDate";

this.PanelTeamMainTenance.Visibility = Visibility.Visible;

break;

case "SendEmail":

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities cvDatabse = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<ModelCommonUse> customerShortInfo = this.customerInfo();

this.Dispatcher.BeginInvoke(

new Action(() =>

{

this.sendMailUsers.ItemsSource = customerShortInfo;

}));

}

})).Start();

this.Option = "OneByOne";

this.PanelSendMail.Visibility = Visibility.Visible;

break;

case "Database":

this.PanelDatabaseBackupRestore.Visibility = Visibility.Visible;

break;

case "Setting":

this.PanelSetting.Visibility = Visibility.Visible;

break;

case "CustomerStatusReset":

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities cvDatabse = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<CustomerAccount> logginCustomers = new ObservableCollection<CustomerAccount>(cvDatabse.CustomerAccounts.Where(x => x.Status == true));

this.Dispatcher.BeginInvoke(

new Action(() =>

{

this.userResGrid.ItemsSource = logginCustomers;

}));

}

})).Start();

new Task(

new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities cvDatabse = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<TeamAccount> logginCustomers = new ObservableCollection<TeamAccount>(cvDatabse.TeamAccounts.Where(x => x.Status == true));

this.Dispatcher.BeginInvoke(

new Action(() =>

{

this.TeamResGrid.ItemsSource = logginCustomers;

}));

}

})).Start();

this.PanelStatusReset.Visibility = Visibility.Visible;

break;

default:

break;

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog(string.Format("Menu >> {0} on Click", obj.ToString()), ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message,CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

}

}

#endregion

#region counterInfoViewer

private void counterInfoViewerStateChanged(object sender, Telerik.Windows.RadRoutedEventArgs e)

{

Telerik.Windows.Controls.RadTileViewItem radItem = e.OriginalSource as Telerik.Windows.Controls.RadTileViewItem;

if (radItem != null)

{

//CommonUse.CounterStatues tempCounterStates = radItem.DataContext as CommonUse.CounterStatues;

//if (tempCounterStates != null)

//{

// tempCounterStates.State = radItem.TileState;

//}

}

}

#endregion

#region Panel Employee Login

public ICommand EmployeeLoginCommand

{

get { return new ReplayCommand(new Action<object>(this.employeeLogin\_Click)); }

}

/// <summary>

/// Employee Login Operation is here.

/// </summary>

/// <param name="obj">Type of ModelEmployee</param>

private void employeeLogin\_Click(object obj)

{

this.EmployeeLogin.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj!=null)

{

ModelEmployee employeeLoginInfo = obj as ModelEmployee;

if (!String.IsNullOrEmpty(employeeLoginInfo.Name) && !String.IsNullOrEmpty(employeeLoginInfo.Password))

{

this.LoginBusyIndicator.IsBusy = true;

Task LoginTask = new Task(new Action(() =>

{

using (Cafeteria\_Vernier\_dbEntities CVDatabase= new Cafeteria\_Vernier\_dbEntities())

{

ViewEmployPermissions defultParmissions = new ViewEmployPermissions();

LoginEmployee = null;

// Query in the database and get employee information and his all permissions.

LoginEmployee = (from employeeInfo in CVDatabase.Employees.ToList()

where employeeInfo.EmployeeID.Trim().Equals(employeeLoginInfo.Name, StringComparison.InvariantCultureIgnoreCase) && employeeInfo.Password.Trim().Equals(employeeLoginInfo.Password)

select new ModelEmployee { UserImage=employeeInfo.UserImage, Address = employeeInfo.Address, Name = employeeInfo.EmployeeID, Password = employeeInfo.Password, PhoneNmber = employeeInfo.Phone,

Permissions = (new ObservableCollection<ModelEmployPermissions>(new ObservableCollection<ModelEmployPermissions>

(from employeeParmisionbd in employeeInfo.EmployeePermissions join employeeParmissions in defultParmissions

on employeeParmisionbd.Privilege.Trim() equals employeeParmissions.Item

orderby employeeParmissions.Priority

select new ModelEmployPermissions { Item = employeeParmissions.Item, Permission = employeeParmissions.Permission, Setting = employeeParmissions.Setting, ImagePath = employeeParmissions.ImagePath, KeboardShortcut = employeeParmissions.KeboardShortcut, Priority = employeeParmissions.Priority, ScreenTip = employeeParmissions.ScreenTip, SupperTip = employeeParmissions.SupperTip })

.Distinct(new ParmissionIequality()))) }).FirstOrDefault();

if (LoginEmployee != null)

{

this.Dispatcher.Invoke(new Action(() =>

{

//Build the main menu

this.MainMenu.ItemsSource = LoginEmployee.Permissions.Where(x => x.Priority != 0 && x.Setting == null).OrderBy(x => x.Priority);

var yy = LoginEmployee.Permissions.Where(x => x.Item.Equals("Setting") && x.Setting != null).OrderBy(x => x.Priority);

this.SettingMenu.ItemsSource = yy;

this.mainMenuClick("Home");

this.MainMenu.Visibility = Visibility.Visible;

this.LoginBusyIndicator.IsBusy = false;

this.LoginBusyIndicator.Visibility = Visibility.Hidden;

this.PanelLogerInfo.Visibility = Visibility.Visible;

}));

}

else

{

this.Dispatcher.Invoke(new Action(() =>

{

Mouse.OverrideCursor = null;

DXMessageBox.Show("Username or Password is not correct.", CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Exclamation);

this.LoginBusyIndicator.IsBusy = false;

}), DispatcherPriority.Normal);

}

}

}));

LoginTask.Start();

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show("Username and Password are empty", CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show("Username and Password are empty",CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Error);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Login Button Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message,CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.EmployeeLogin.IsEnabled = true;

}

}

#endregion

#region Panel LogerInfo or logout

public ICommand LogoutCommand

{

get { return new ReplayCommand(new Action<object>(this.logout\_Click)); }

}

/// <summary>

/// Employee Logout

/// </summary>

/// <param name="obj"></param>

private void logout\_Click(object obj)

{

this.EmployeeLogout.IsEnabled = false;

try

{

if (obj != null)

{

if (DXMessageBox.Show("Logout ! are you sure ?", CvVariables.SOFTWARE\_NAME, MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

Mouse.OverrideCursor = Cursors.Wait;

this.MainMenu.Visibility = Visibility.Hidden;

this.hidePanels();

this.LoginBusyIndicator.Visibility = Visibility.Visible;

this.PanelLogerInfo.Visibility = Visibility.Hidden;

this.EmployeeLoginUsername.Focus();

ModelEmployee loginInfo = obj as ModelEmployee;

loginInfo.Name = loginInfo.Password = string.Empty;

this.LoginEmployee = null;

}

else

{

return;

}

}

else

{

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0,0], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Employee Logout Click", ErrorException);

DXMessageBox.Show(ErrorException.Message,CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.EmployeeLogout.IsEnabled = true;

}

}

#endregion

#region Panel New Customer

public ICommand NewCustomerCommand

{

get { return new ReplayCommand(new Action<object>(this.newCustomer\_Click)); }

}

public ICommand UpdateCustomerCommand

{

get { return new ReplayCommand(new Action<object>(this.updateCustomer\_Click)); }

}

public ICommand DeleteCustomerCommand

{

get { return new ReplayCommand(new Action<object>(this.deleteCustomer\_Click)); }

}

public ICommand BrowseCustomerCommand

{

get { return new ReplayCommand(new Action<object>(this.browseCustomer\_Click)); }

}

public ICommand WebCamCustomerCommand

{

get { return new ReplayCommand(new Action<object>(this.webCamCustomer\_Click)); }

}

/// <summary>

/// Add new Customer

/// </summary>

/// <param name="obj"> No parameter need</param>

private void newCustomer\_Click(object obj)

{

this.CustomerNew.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

ModelCustomer newCustomer = new ModelCustomer();

newCustomer.JoinDate = DateTime.Today;

(this.CustomerList.ItemsSource as ObservableCollection<ModelCustomer>).Add(newCustomer);

this.CustomerList.SelectedIndex = this.CustomerList.Items.IndexOf(newCustomer);

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("New Customer Click", ErrorException);

DXMessageBox.Show(ErrorException.Message, CvVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.CustomerNew.IsEnabled = true;

}

}

/// <summary>

/// Update or insert new Customer

/// </summary>

/// <param name="obj">ListBox Selected Item</param>

private void updateCustomer\_Click(object obj)

{

this.CustomerUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj != null)

{

this.CustomerUserName.GetBindingExpression(TextEdit.TextProperty);

this.CustomerPassword.GetBindingExpression(PasswordBoxEdit.PasswordProperty);

this.CustomerName.GetBindingExpression(TextEdit.TextProperty);

this.CustomerPhone.GetBindingExpression(TextEdit.TextProperty);

this.CustomerEmail.GetBindingExpression(TextEdit.TextProperty);

this.getValidationError(this.CustomerUserName, this.CustomerPassword, this.CustomerPhone, this.CustomerPhone, this.CustomerName, this.CustomerEmail);

using (Cafeteria\_Vernier\_dbEntities CVDatabase= new Cafeteria\_Vernier\_dbEntities())

{

ModelCustomer selectedCustomer = obj as ModelCustomer;

var customerExits = CVDatabase.CustomerInformations.FirstOrDefault(x=>x.UserID.Equals(selectedCustomer.UserName));

if (customerExits!=null)

{

if (this.LoginEmployee.Permissions.AsParallel().FirstOrDefault(x=>x.Item == ("EditCustomer"))!=null)

{

Mouse.OverrideCursor = null;

if (DXMessageBox.Show(CvVariables.ERROR\_MESSAGES[1, 8], CvVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

Mouse.OverrideCursor = Cursors.Wait;

customerExits.Address = selectedCustomer.Address;

customerExits.CustomerAccount.Password = selectedCustomer.Password;

customerExits.CustomerAccount.Minutes = selectedCustomer.Minutes;

customerExits.Email = selectedCustomer.Email;

customerExits.JoinDate = selectedCustomer.JoinDate;

customerExits.Logo = selectedCustomer.Image;

customerExits.NationalID = selectedCustomer.NationalID;

customerExits.Phone = selectedCustomer.Phone;

CVDatabase.SaveChanges();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 2], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

return;

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 1], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Hand);

}

}

else

{

CVDatabase.CustomerInformations.AddObject(

new CustomerInformation

{

Address = selectedCustomer.Address,

Email = selectedCustomer.Email,

JoinDate = selectedCustomer.JoinDate,

Logo = selectedCustomer.Image,

Name = selectedCustomer.Name,

NationalID = selectedCustomer.NationalID,

Phone = selectedCustomer.Phone,

UserID = selectedCustomer.UserName,

CustomerAccount = new CustomerAccount

{

Counternumber = 0,

Minutes = selectedCustomer.Minutes,

Password = selectedCustomer.Password,

Status = false

}

});

CVDatabase.SaveChanges();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 3], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 1], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Hand);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Customer Update Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.CustomerUpdate.IsEnabled = true;

}

}

/// <summary>

/// Delete a Customer

/// </summary>

/// <param name="obj">ListBox Selected Item</param>

private void deleteCustomer\_Click(object obj)

{

this.CustomerDelete.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj != null)

{

using (Cafeteria\_Vernier\_dbEntities CVDatabase= new Cafeteria\_Vernier\_dbEntities())

{

ModelCustomer deleteCustomerInfo = obj as ModelCustomer;

CVDatabase.CustomerInformations.DeleteObject(CVDatabase.CustomerInformations.First(x => x.UserID.Equals(deleteCustomerInfo.UserName)));

CVDatabase.SaveChanges();

(this.CustomerList.ItemsSource as ObservableCollection<ModelCustomer>).Remove(deleteCustomerInfo);

this.selectListBoxFirstItem(this.CustomerList);

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 5], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 1], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Hand);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Customer Delete Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.CustomerDelete.IsEnabled = true;

}

}

/// <summary>

/// Browse a Customer Image

/// </summary>

/// <param name="obj">ListBox Selected Item</param>

private void browseCustomer\_Click(object obj)

{

this.CustomerBrowse.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj != null)

{

(obj as ModelCustomer).Image = this.selectImageFromFile();

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 4], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Exclamation);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Customer Browse Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.CustomerBrowse.IsEnabled = true;

}

}

/// <summary>

/// Snapshot from WebCam

/// </summary>

/// <param name="obj">ListBox Selected Item</param>

private void webCamCustomer\_Click(object obj)

{

this.CustomerWebCam.IsEnabled = false;

try

{

if (obj != null)

{

byte[] imageInBytes = null;

imageInBytes = WebCamWindow.CaptureImage();

Mouse.OverrideCursor = Cursors.Wait;

if (imageInBytes != null)

{

(obj as ModelCustomer).Image = imageInBytes;

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 4], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Exclamation);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Customer WebCam Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.CustomerWebCam.IsEnabled = true;

}

}

#endregion

#region Panel Account Recharge

public ICommand AccountRechargeCommand

{

get { return new ReplayCommand(new Action<object>(this.accountRechargeClick)); }

}

/// <summary>

/// Update Customer or Team Account

/// Insert into Customer and Team Account recharge history

/// </summary>

/// <param name="obj">

/// obj[0] = ModelBillConfig

/// obj[1] = ModelCustomer

/// obj[2] = ModelTeamInfo

/// obj[3] = CustomerInfo.IsChecked

/// </param>

private void accountRechargeClick(object obj)

{

this.RechareUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is ArrayList)

{

ArrayList dataList= obj as ArrayList;

ModelBillConfig billInfo = dataList[0] as ModelBillConfig;

ModelCustomer customerInfo = dataList[1] as ModelCustomer;

ModelTeamInfo teamInfo = dataList[2] as ModelTeamInfo;

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

if ((bool)dataList[3])

{

var userAccountInfo = cvDatabase.CustomerAccounts.First(x => x.UserID.Equals(customerInfo.Name));

userAccountInfo.Minutes += Convert.ToInt32(billInfo.Minutes);

cvDatabase.AddToUserRechargeHistories(

new UserRechargeHistory

{

AutoInc = default(long),

bill = Convert.ToInt32(billInfo.Amount),

DateWithTime = DateTime.Today,

EmployeeID = this.LoginEmployee.Name,

Minutes = Convert.ToInt32(billInfo.Minutes),

UserID = customerInfo.Name

});

customerInfo.Minutes += Convert.ToInt32(billInfo.Minutes);

}

else

{

var teamAccountInfo = cvDatabase.TeamAccounts.First(x => x.Name.Equals(teamInfo.Name));

teamAccountInfo.Minutes += Convert.ToInt32(billInfo.Minutes);

cvDatabase.AddToTeamRechargeHistories(

new TeamRechargeHistory

{

AutoInc = default(long),

bill = Convert.ToInt32(billInfo.Amount),

DateWithTime = DateTime.Today,

EmployeeID = this.LoginEmployee.Name,

Minutes = Convert.ToInt32(billInfo.Minutes),

Name = teamInfo.Name

});

teamInfo.Minutes += Convert.ToInt32(billInfo.Minutes);

}

var toDayCash = cvDatabase.Cashes.First(x => x.EntryDate == DateTime.Today);

toDayCash.Amount += Convert.ToDecimal(billInfo.Amount);

cvDatabase.SaveChanges();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0,2],CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Information);

}

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Account Recharge", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message,CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.RechareUpdate.IsEnabled = true;

}

}

#endregion

#region Panel Recharges history view

public ICommand SearchRechargeHistoryCommand

{

get { return new ReplayCommand(new Action<object>(this.searchRechargeHistoryClick)); }

}

public ICommand DeleteRechareHistoryCommand

{

get { return new ReplayCommand(new Action<object>(this.deleteRechageHistoryClick)); }

}

public ICommand DeleteAllRechageHistoryClick

{

get { return new ReplayCommand(new Action<object>(this.deleteAllRechageHistoryClick)); }

}

/// <summary>

/// Search recharge History

/// </summary>

/// <param name="obj">

/// obj[0] = isCustomer(bool)

/// obj[1] = Is by Name (bool)

/// obj[2] = UserName

/// obj[3] = TeamName

/// obj[4] = date One

/// obj[5] = date two

/// </param>

private void searchRechargeHistoryClick(object obj)

{

this.ResHiSearch.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is ArrayList)

{

ArrayList datalist = obj as ArrayList;

DateTime firstDate=(DateTime)datalist[4];

DateTime secondDate=(DateTime)datalist[5];

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

if ((bool)datalist[0]) // Is Customer

{

IQueryable<UserRechargeHistory> queryUserRechargeHistory = null;

this.ResHisGridView.Columns.OfType<GridViewDataColumn>().First(x => x.Name == "GridDataColumnName").DataMemberBinding = new Binding("UserID");

if ((bool)datalist[1]) // Is by Name

{

string userName = datalist[2].ToString();

switch (this.Option)

{

case "ByDate":

queryUserRechargeHistory = cvDatabase.UserRechargeHistories.Where(x => x.DateWithTime == firstDate && x.UserID == userName);

break;

case "BetweenTwoDate":

queryUserRechargeHistory = cvDatabase.UserRechargeHistories.Where(x => x.DateWithTime >= firstDate && x.DateWithTime <= secondDate && x.UserID == userName);

break;

case "All":

queryUserRechargeHistory = cvDatabase.UserRechargeHistories.Where(x => x.UserID == userName);

break;

default:

break;

}

}

else

{

switch (this.Option)

{

case "ByDate":

queryUserRechargeHistory = cvDatabase.UserRechargeHistories.Where(x => x.DateWithTime == firstDate);

break;

case "BetweenTwoDate":

queryUserRechargeHistory = cvDatabase.UserRechargeHistories.Where(x => x.DateWithTime >= firstDate && x.DateWithTime <= secondDate);

break;

case "All":

queryUserRechargeHistory = cvDatabase.UserRechargeHistories;

break;

default:

break;

}

}

this.ResHisGridView.ItemsSource =new ObservableCollection<UserRechargeHistory>(queryUserRechargeHistory);

}

else

{

IQueryable<TeamRechargeHistory> queryTeamRechareHistory = null;

this.ResHisGridView.Columns.OfType<GridViewDataColumn>().First(x => x.Name == "GridDataColumnName").DataMemberBinding = new Binding("Name");

if ((bool)datalist[1]) // Is By Name

{

string teamName = datalist[3].ToString();

switch (this.Option)

{

case "ByDate":

queryTeamRechareHistory = cvDatabase.TeamRechargeHistories.Where(x => x.DateWithTime == firstDate && x.Name == teamName);

break;

case "BetweenTwoDate":

queryTeamRechareHistory = cvDatabase.TeamRechargeHistories.Where(x => x.DateWithTime >= firstDate && x.DateWithTime <= secondDate && x.Name == teamName);

break;

case "All":

queryTeamRechareHistory = cvDatabase.TeamRechargeHistories.Where(x => x.Name == teamName);

break;

default:

break;

}

}

else

{

switch (this.Option)

{

case "ByDate":

queryTeamRechareHistory = cvDatabase.TeamRechargeHistories.Where(x => x.DateWithTime == firstDate);

break;

case "BetweenTwoDate":

queryTeamRechareHistory = cvDatabase.TeamRechargeHistories.Where(x => x.DateWithTime >= firstDate && x.DateWithTime <= secondDate);

break;

case "All":

queryTeamRechareHistory = cvDatabase.TeamRechargeHistories;

break;

default:

break;

}

}

this.ResHisGridView.ItemsSource = new ObservableCollection<TeamRechargeHistory>(queryTeamRechareHistory);

}

}

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Recharge History Search", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message,CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.ResHiSearch.IsEnabled = true;

}

}

/// <summary>

/// Delete Selected Items

/// </summary>

/// <param name="obj">

/// obj[0]=DataGrid.SelectedItems

/// obj[1]=Is Customer

/// </param>

private void deleteRechageHistoryClick(object obj)

{

this.ResHisDelete.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is ArrayList)

{

ArrayList dataList = obj as ArrayList;

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

if ((bool)dataList[1])

{

List<UserRechargeHistory> historyList = new List<UserRechargeHistory>();

foreach (var singleListory in dataList[0] as IEnumerable)

{

historyList.Add(singleListory as UserRechargeHistory);

}

foreach (UserRechargeHistory loginHistory in historyList)

{

cvDatabase.UserRechargeHistories.DeleteObject(cvDatabase.UserRechargeHistories.First(x=>x.DateWithTime == loginHistory.DateWithTime && x.EmployeeID.Equals(loginHistory.EmployeeID) && x.UserID.Equals(loginHistory.UserID)));

(this.ResHisGridView.ItemsSource as ObservableCollection<UserRechargeHistory>).Remove(loginHistory);

}

}

else

{

List<TeamRechargeHistory> historyList = new List<TeamRechargeHistory>();

foreach (var singleListory in dataList[0] as IEnumerable)

{

historyList.Add(singleListory as TeamRechargeHistory);

}

foreach (TeamRechargeHistory loginHistory in historyList)

{

cvDatabase.TeamRechargeHistories.DeleteObject(cvDatabase.TeamRechargeHistories.First(x=>x.DateWithTime == loginHistory.DateWithTime && x.EmployeeID.Equals(loginHistory.EmployeeID) && x.Name.Equals(loginHistory.Name)));

(this.ResHisGridView.ItemsSource as ObservableCollection<TeamRechargeHistory>).Remove(loginHistory);

}

}

cvDatabase.SaveChanges();

}

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 5], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Recharge History Delete", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message,CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.ResHisDelete.IsEnabled = true;

}

}

/// <summary>

/// Delete all History

/// </summary>

/// <param name="obj">

/// obj[0] = DataGrid.itemSource

/// objj[1] = Is Customer

/// </param>

private void deleteAllRechageHistoryClick(object obj)

{

this.ResHisDeleteAll.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is ArrayList)

{

ArrayList dataList = obj as ArrayList;

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

if ((bool)dataList[1])

{

ObservableCollection<UserRechargeHistory> userRecharges = new ObservableCollection<UserRechargeHistory>();

foreach (UserRechargeHistory userRecharge in dataList[0] as ObservableCollection<UserRechargeHistory>)

{

userRecharges.Add(userRecharge);

}

foreach (UserRechargeHistory rechargeRecord in userRecharges)

{

cvDatabase.UserRechargeHistories.DeleteObject(cvDatabase.UserRechargeHistories.First(x=>x.UserID.Equals(rechargeRecord.UserID) && x.DateWithTime == rechargeRecord.DateWithTime && x.EmployeeID.Equals(rechargeRecord.EmployeeID) ));

(this.ResHisGridView.ItemsSource as ObservableCollection<UserRechargeHistory>).Remove(rechargeRecord);

}

}

else

{

ObservableCollection<TeamRechargeHistory> teamRecharges = new ObservableCollection<TeamRechargeHistory>();

foreach (TeamRechargeHistory userRecharge in dataList[0] as ObservableCollection<TeamRechargeHistory>)

{

teamRecharges.Add(userRecharge);

}

foreach (TeamRechargeHistory rechargeRecord in teamRecharges)

{

cvDatabase.TeamRechargeHistories.DeleteObject(cvDatabase.TeamRechargeHistories.First(x=>x.Name.Equals(rechargeRecord.Name) && x.DateWithTime == rechargeRecord.DateWithTime && x.EmployeeID.Equals(rechargeRecord.EmployeeID)));

(this.ResHisGridView.ItemsSource as ObservableCollection<TeamRechargeHistory>).Remove(rechargeRecord);

}

}

cvDatabase.SaveChanges();

}

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 5], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Recharge History Delete All", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message);

}

finally

{

Mouse.OverrideCursor = null;

this.ResHisDeleteAll.IsEnabled = true;

}

}

#endregion

#region Panel Cutomer Login History

public ICommand CustLoginHisSearchCommand

{

get { return new ReplayCommand(new Action<object>(this.custLoginHisSearchClick)); }

}

public ICommand CustLoginHisDeleteCommand

{

get { return new ReplayCommand(new Action<object>(this.custLoginHisDeleteClick)); }

}

public ICommand CustLoginHisDeleteAllCommand

{

get { return new ReplayCommand(new Action<object>(this.custLoginHisDeleteAllClick)); }

}

/// <summary>

/// Search Login History

/// </summary>

/// <param name="obj">

/// obj[0] = Name

/// obj[1]= IsName

/// obj[2] = First Date

/// obj[3] = Second Date

/// obj[4] = Third Date

/// </param>

private void custLoginHisSearchClick(object obj)

{

this.LogHisSearch.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is ArrayList)

{

ArrayList dataList = obj as ArrayList;

DateTime firstDate = (DateTime)dataList[2];

DateTime secondDate = (DateTime)dataList[3];

DateTime thirdDate = (DateTime)dataList[4];

IQueryable<UserLoginHistory> searchQuery = null;

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

if ((bool)dataList[1])

{

string useName = dataList[0].ToString();

switch (this.Option)

{

case "ByDate":

searchQuery = cvDatabase.UserLoginHistories.Where(x => x.StratTime == firstDate && x.UserID == useName);

break;

case "BetweenTwoDate":

searchQuery = cvDatabase.UserLoginHistories.Where(x => x.StratTime == secondDate && x.StratTime == thirdDate && x.UserID == useName);

break;

case "All":

searchQuery = cvDatabase.UserLoginHistories.Where(x => x.UserID == useName);

break;

default:

break;

}

}

else

{

switch (this.Option)

{

case "ByDate":

searchQuery = cvDatabase.UserLoginHistories.Where(x => x.StratTime == firstDate);

break;

case "BetweenTwoDate":

searchQuery = cvDatabase.UserLoginHistories.Where(x => x.StratTime == secondDate && x.StratTime == thirdDate);

break;

case "All":

searchQuery = cvDatabase.UserLoginHistories;

break;

default:

break;

}

}

this.LogHisGridView.ItemsSource = new ObservableCollection<ModelUserLogin>(searchQuery.Select(x => new ModelUserLogin {MuniteUses = x.EndTime.Value.Minute-x.StratTime.Minute, CounterNumber=x.CounterNumber, StratTime=x.StratTime, TeamName=x.TeamName, UserID=x.UserID }));

}

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Login History Search", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.LogHisSearch.IsEnabled = true;

}

}

/// <summary>

/// Delete History

/// </summary>

/// <param name="obj"> GridView.SelectedItems </param>

private void custLoginHisDeleteClick(object obj)

{

this.LogHisDelete.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

ObservableCollection<ModelUserLogin> loginHistorys = new ObservableCollection<ModelUserLogin>();

foreach (ModelUserLogin loginHistory in obj as IEnumerable)

{

loginHistorys.Add(loginHistory);

}

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

foreach (ModelUserLogin loginHistory in loginHistorys)

{

cvDatabase.UserLoginHistories.DeleteObject(cvDatabase.UserLoginHistories.First(x=>x.StratTime == loginHistory.StratTime && x.UserID.Equals(loginHistory.UserID)));

(this.LogHisGridView.ItemsSource as ObservableCollection<ModelUserLogin>).Remove(loginHistory);

}

cvDatabase.SaveChanges();

}

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 5], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Login History Delete", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.LogHisDelete.IsEnabled = true;

}

}

/// <summary>

/// Delete All History

/// </summary>

/// <param name="obj">DataGrid.ItemSource</param>

private void custLoginHisDeleteAllClick(object obj)

{

this.LogHisDeleteAll.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

ObservableCollection<ModelUserLogin> userLoginHistorys = new ObservableCollection<ModelUserLogin>();

foreach (ModelUserLogin loginHistory in userLoginHistorys)

{

userLoginHistorys.Add(loginHistory);

}

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

foreach (ModelUserLogin loginHistory in userLoginHistorys)

{

cvDatabase.UserLoginHistories.DeleteObject(cvDatabase.UserLoginHistories.First(x=>EntityFunctions.TruncateTime(x.StratTime) == loginHistory.StratTime && x.UserID.Equals(loginHistory.UserID)));

(this.LogHisGridView.ItemsSource as ObservableCollection<ModelUserLogin>).Remove(loginHistory);

}

cvDatabase.SaveChanges();

}

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 5], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Login History Delete All", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.LogHisDeleteAll.IsEnabled = true;

}

}

#endregion

#region Panel Cash View and Edit

public ICommand SearchCashCommand

{

get { return new ReplayCommand(new Action<object>(this.searchCashClick)); }

}

public ICommand UpdateCashCommand

{

get { return new ReplayCommand(new Action<object>(this.updateCashClick)); }

}

/// <summary>

/// Search Cash by Date

/// </summary>

/// <param name="obj"> Date time</param>

private void searchCashClick(object obj)

{

this.CashSearch.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait; ;

try

{

DateTime selectedDate = (DateTime)obj;

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

this.CashAmount.Value =Convert.ToDouble(cvDatabase.Cashes.FirstOrDefault(x => x.EntryDate == selectedDate).Amount);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Cash Search", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.CashSearch.IsEnabled = true;

}

}

/// <summary>

/// Update Cash

/// </summary>

/// <param name="obj">

/// obj[0]=dateTime

/// obj[1]=Amount

/// </param>

private void updateCashClick(object obj)

{

this.CashUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait; ;

try

{

if (obj is ArrayList)

{

ArrayList dataList = obj as ArrayList;

DateTime selectedDate=(DateTime)dataList[0];

double cashAmount= (double)dataList[1];

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

var selectedCash = cvDatabase.Cashes.FirstOrDefault(x => x.EntryDate == selectedDate);

selectedCash.Amount = Convert.ToDecimal(cashAmount);

cvDatabase.SaveChanges();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0,2], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Cash Update", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.CashUpdate.IsEnabled = true;

}

}

#region Olde Code

/// <summary>

/// Search Button Click

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void CashViewButtonSearchClick(object sender, System.Windows.RoutedEventArgs e)

{

//this.CashViewButtonSearch.IsEnabled = false;

//Mouse.OverrideCursor = Cursors.Wait;

//try

//{

// this.CashViewtextBolockAmount.Text =(from cashtable in new CvDataClassesDataContext().CV\_Cashes where cashtable.EntryDate.Equals(this.CashViewDate.SelectedDate) select new { cashtable.Amount }).Single().Amount.ToString();

//}

//catch (Exception error)

//{

// Mouse.OverrideCursor = null;

// DXMessageBox.Show(error.Message, CvVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

//}

//finally

//{

// Mouse.OverrideCursor = null;

// this.CashViewButtonSearch.IsEnabled=true;

//}

}

#endregion

#endregion

#region Panel Cash History

public ICommand SearchCashHisotyCommand

{

get { return new ReplayCommand(new Action<object>(this.searchCashHistoryClick)); }

}

public ICommand PrintCashHistoryCommand

{

get { return new ReplayCommand(new Action<object>(this.printCashHistoryClick)); }

}

/// <summary>

/// Search For Cash History

/// </summary>

/// <param name="obj">

/// obj[0]=FirstDate

/// obj[1] = Second Date

/// obj[2] = Thread Date

/// </param>

private void searchCashHistoryClick(object obj)

{

this.CashHistorySearch.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is ArrayList)

{

ArrayList dataList = obj as ArrayList;

IQueryable<Cash> searchQuery = null;

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

switch (this.Option)

{

case "ByDate":

DateTime firstDate = (DateTime)dataList[0];

searchQuery = cvDatabase.Cashes.Where(x => x.EntryDate == firstDate);

break;

case "BetweenTwoDate":

DateTime secondDate=(DateTime)dataList[1];

DateTime thirdDate=(DateTime)dataList[2];

searchQuery = cvDatabase.Cashes.Where(x => x.EntryDate >= secondDate && x.EntryDate <= thirdDate);

break;

case "All":

searchQuery =cvDatabase.Cashes;

break;

default:

break;

}

this.CashHistoryGrid.ItemsSource = new ObservableCollection<Cash>(searchQuery.Select(x=>x));

}

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Cash History Search", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message,CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.CashHistorySearch.IsEnabled = true;

}

}

/// <summary>

/// Print History

/// </summary>

/// <param name="obj"></param>

private void printCashHistoryClick(object obj)

{

this.CashHistoryPrint.IsEnabled = false;

try

{

CashHistoryReport newCashHistoryReport = new CashHistoryReport();

newCashHistoryReport.CashHistoryBindingSource.DataSource = obj as IEnumerable;

PrintHelper.ShowPrintPreviewDialog(this, newCashHistoryReport);

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Cash History Print", ErrorException);

DXMessageBox.Show(ErrorException.Message,CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Error);

}

finally

{

this.CashHistoryPrint.IsEnabled = true;

}

}

#endregion

#region Panel Business Summary

public ICommand SearchSummaryCommand

{

get { return new ReplayCommand(new Action<object>(this.searchSummaryClick)); }

}

/// <summary>

/// Search Daily Summary

/// </summary>

/// <param name="obj">DateTime</param>

private void searchSummaryClick(object obj)

{

this.SummarySearch.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

DateTime selectedDate = (DateTime)obj;

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

double customerMinutes=0, teamMinutes=0;

var cashByDate= cvDatabase.Cashes.FirstOrDefault(x => x.EntryDate == selectedDate);

this.SummaryCash.Text = cashByDate != null ? cashByDate.Amount.ToString() : "0";

var customerQuery = cvDatabase.UserRechargeHistories.Where( x => x.DateWithTime == selectedDate);

var teamQuery = cvDatabase.TeamRechargeHistories.Where(x => x.DateWithTime == selectedDate);

if (customerQuery.Count()>0)

{

customerMinutes = customerQuery.Sum(x => x.Minutes);

}

if (teamQuery.Count()>0)

{

teamMinutes = teamQuery.Sum(x => x.Minutes);

}

this.SummaryMinutes.Text = (customerMinutes + teamMinutes).ToString();

this.SummaryTotalLogin.Text = (from totalLogin in cvDatabase.UserLoginHistories where EntityFunctions.TruncateTime(totalLogin.StratTime) == selectedDate select totalLogin).Count().ToString();

//cvDatabase.UserLoginHistories.Where(x => x.StratTime == selectedDate).Count().ToString();

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Summary Search", ErrorException);

DXMessageBox.Show(ErrorException.Message,CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.SummarySearch.IsEnabled = true;

}

}

#endregion

#region Panel New Team

public ICommand NewTeamCommand

{

get { return new ReplayCommand(new Action<object>(this.newTeam\_Click)); }

}

public ICommand UpdateTeamCommand

{

get { return new ReplayCommand(new Action<object>(this.updateTeam\_Click)); }

}

public ICommand DeleteTeamCommand

{

get { return new ReplayCommand(new Action<object>(this.deleteTeam\_Click)); }

}

public ICommand BrowseTeamCommand

{

get { return new ReplayCommand(new Action<object>(this.browseTeam\_Click)); }

}

public ICommand WebCamTeamCommand

{

get { return new ReplayCommand(new Action<object>(this.webCamTeam\_Click)); }

}

public ICommand TeamMemberAddCommand

{

get { return new ReplayCommand(new Action<object>(this.memberAdd\_Click)); }

}

public ICommand TeamMemberRemoveCommand

{

get { return new ReplayCommand(new Action<object>(this.memberRemove\_Click)); }

}

/// <summary>

/// Add New Team

/// </summary>

/// <param name="obj"></param>

private void newTeam\_Click(object obj)

{

this.TeamNew.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

ModelTeamInfo newTeamInfo = new ModelTeamInfo();

newTeamInfo.TeamMemberList = new ObservableCollection<ModelCommonUse>();

(this.TeamGridView.ItemsSource as ObservableCollection<ModelTeamInfo>).Add(newTeamInfo);

this.TeamGridView.Rebind();

this.TeamGridView.SelectedItem = this.TeamGridView.Items[this.TeamGridView.Items.IndexOf(newTeamInfo)];

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Team New Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.TeamNew.IsEnabled = true;

}

}

/// <summary>

/// Update or insert Team Information

/// </summary>

/// <param name="obj"></param>

private void updateTeam\_Click(object obj)

{

this.TeamUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj!=null)

{

this.TeamName.GetBindingExpression(TextEdit.TextProperty);

this.TeamAdminName.GetBindingExpression(TextEdit.TextProperty);

this.getValidationError(this.TeamName,this.TeamAdminName);

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

ModelTeamInfo selectedTeamInfo = obj as ModelTeamInfo;

var teamExist = cvDatabase.Teams.FirstOrDefault(x => x.Name.Equals(selectedTeamInfo.Name));

if (teamExist!=null)

{

Mouse.OverrideCursor = null;

if (DXMessageBox.Show(CvVariables.ERROR\_MESSAGES[1, 8], CvVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

teamExist.AdminName = selectedTeamInfo.AdminName;

teamExist.JoinDate = selectedTeamInfo.JoinDate;

teamExist.Logo = selectedTeamInfo.Image;

teamExist.TeamAccount.Minutes = selectedTeamInfo.Minutes;

foreach (var oldMember in cvDatabase.TeamMembers.Where(x => x.Name.Trim().Equals(selectedTeamInfo.Name)))

{

cvDatabase.TeamMembers.DeleteObject(oldMember);

}

// teamExist.TeamMembers.Clear();

foreach (var newMember in selectedTeamInfo.TeamMemberList)

{

teamExist.TeamMembers.Add(new TeamMember{UserID = newMember.UserName, AutoInc = default(long), Name = selectedTeamInfo.Name});

}

cvDatabase.SaveChanges();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 2], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

return;

}

}

else

{

Team newTeam = new Team

{

JoinDate = selectedTeamInfo.JoinDate,

Logo = selectedTeamInfo.Image,

Name = selectedTeamInfo.Name,

AdminName = selectedTeamInfo.AdminName,

TeamAccount = new TeamAccount { Minutes=selectedTeamInfo.Minutes, Status=false, EntryDate=DateTime.Today }

};

foreach (var newMember in selectedTeamInfo.TeamMemberList)

{

newTeam.TeamMembers.Add(new TeamMember

{

AutoInc = default(long),

UserID = newMember.UserName

});

}

cvDatabase.Teams.AddObject(newTeam);

cvDatabase.SaveChanges();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 3], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 4], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Exclamation);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Update Or Insert Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.TeamUpdate.IsEnabled = true;

}

}

/// <summary>

/// Delete a team information

/// </summary>

/// <param name="obj"></param>

private void deleteTeam\_Click(object obj)

{

this.TeamDelete.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj!=null)

{

ModelTeamInfo deleteedTeam = obj as ModelTeamInfo;

using (Cafeteria\_Vernier\_dbEntities cvDatbase= new Cafeteria\_Vernier\_dbEntities())

{

cvDatbase.Teams.DeleteObject(cvDatbase.Teams.First(x=>x.Name.Equals(deleteedTeam.Name)));

cvDatbase.SaveChanges();

(this.TeamGridView.ItemsSource as ObservableCollection<ModelTeamInfo>).Remove(deleteedTeam);

this.TeamGridView.Rebind();

this.selectGridViewFirstItem(this.TeamGridView);

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 5], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 4], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Exclamation);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Team Delete Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.TeamDelete.IsEnabled = true;

}

}

/// <summary>

/// Browse image for a Team

/// </summary>

/// <param name="obj"></param>

private void browseTeam\_Click(object obj)

{

this.TeamBrowes.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj!=null)

{

(obj as ModelTeamInfo).Image = this.selectImageFromFile();

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 4], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Exclamation);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Team Browse Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.TeamBrowes.IsEnabled = true;

}

}

/// <summary>

/// Snapshot from webcam

/// </summary>

/// <param name="obj"></param>

private void webCamTeam\_Click(object obj)

{

this.TeamWebcam.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj != null)

{

byte[] imageInBytes = null;

imageInBytes = WebCamWindow.CaptureImage();

Mouse.OverrideCursor = Cursors.Wait;

if (imageInBytes != null)

{

(obj as ModelTeamInfo).Image = imageInBytes;

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 4], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Exclamation);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Team web cam Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.TeamWebcam.IsEnabled = true;

}

}

/// <summary>

/// Add new Member to the team

/// </summary>

/// <param name="obj"></param>

private void memberAdd\_Click(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj !=null)

{

ModelCommonUse newMember = obj as ModelCommonUse;

var membersList= this.TeamMemberList.ItemsSource as ObservableCollection<ModelCommonUse>;

if (membersList.FirstOrDefault(x=>x.UserName.Trim().Equals(newMember.UserName))==null)

{

membersList.Add(newMember);

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show("This user is already a member.", CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Stop);

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 4], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Exclamation);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Team member add Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

}

}

/// <summary>

/// Remove Member from Member list

/// </summary>

/// <param name="obj"></param>

private void memberRemove\_Click(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj!=null)

{

(this.TeamMemberList.ItemsSource as ObservableCollection<ModelCommonUse>).Remove(obj as ModelCommonUse);

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 4], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Exclamation);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Team member remove click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

}

}

#endregion

#region Panel User Maintenance

public ICommand UserMaintenanceSearchCommand

{

get { return new ReplayCommand(new Action<object>(this.userMaintenanceSearchClick)); }

}

public ICommand UserMiantenanceDeleteCommand

{

get { return new ReplayCommand(new Action<object>(this.userMaintenanceDeleteClick)); }

}

public ICommand UserMiantenanceDeleteAllCommand

{

get { return new ReplayCommand(new Action<object>(this.userMaintenanceDeleteAllClick)); }

}

/// <summary>

/// Search User`s

/// </summary>

/// <param name="obj">

/// obj[0] = Name

/// obj[1] = FirstDate

/// obj[2] = Second Date

/// obj[3] = Third Date

/// obj[4] = Minutes

/// </param>

private void userMaintenanceSearchClick(object obj)

{

this.UserMaintenanceSearch.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is ArrayList)

{

ArrayList dataList = obj as ArrayList;

IQueryable<CustomerInformation> searchQuery = null;

Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities();

switch (this.Option)

{

case "ByName":

string username=(string)dataList[0];

searchQuery = cvDatabase.CustomerInformations.Where(x => x.UserID == username);

break;

case "ByDate":

switch (this.SubOption)

{

case "ByDate":

DateTime joinDateEqual = (DateTime)dataList[1];

searchQuery = cvDatabase.CustomerInformations.Where(x => x.JoinDate == joinDateEqual);

break;

case "Below":

DateTime joinDateBelow = (DateTime)dataList[1];

searchQuery = cvDatabase.CustomerInformations.Where(x => x.JoinDate <= joinDateBelow);

break;

case "TwoDate":

DateTime firstDate = (DateTime)dataList[2];

DateTime seconDate = (DateTime)dataList[3];

searchQuery = cvDatabase.CustomerInformations.Where(x => x.JoinDate <= firstDate && x.JoinDate>=seconDate);

break;

default:

break;

}

break;

case "ByAmount":

var minutes = (double)dataList[4];

switch (this.SubOption)

{

case "Below":

searchQuery = cvDatabase.CustomerInformations.Where(x => x.CustomerAccount.Minutes<=minutes);

break;

case "Equal":

searchQuery = cvDatabase.CustomerInformations.Where(x => x.CustomerAccount.Minutes == minutes);

break;

}

break;

default:

break;

}

this.UserMaintenanceCutomerGridView.ItemsSource = new ObservableCollection<CustomerInformation>(searchQuery);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("User Maintenance Search", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.UserMaintenanceSearch.IsEnabled = true;

}

}

/// <summary>

/// Delete Selected User

/// </summary>

/// <param name="obj">GridView.SelectedItems</param>

private void userMaintenanceDeleteClick(object obj)

{

this.UserMaintenanceDelete.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<CustomerInformation> customerInfos = new ObservableCollection<CustomerInformation>();

foreach (var singleItem in obj as IEnumerable)

{

customerInfos.Add(singleItem as CustomerInformation);

}

foreach (CustomerInformation singleCutomer in customerInfos)

{

cvDatabase.CustomerInformations.DeleteObject(cvDatabase.CustomerInformations.First(x=>x.UserID.Equals(singleCutomer.UserID)));

(this.UserMaintenanceCutomerGridView.ItemsSource as ObservableCollection<CustomerInformation>).Remove(singleCutomer);

}

cvDatabase.SaveChanges();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 5], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("User Maintenance Delete", ErrorException);

DXMessageBox.Show(ErrorException.Message);

}

finally

{

Mouse.OverrideCursor = null;

this.UserMaintenanceDelete.IsEnabled = true;

}

}

/// <summary>

/// Delete All User Info

/// </summary>

/// <param name="obj">Gridview.ItemSource</param>

private void userMaintenanceDeleteAllClick(object obj)

{

this.UserMaintenanceDeleteAll.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<CustomerInformation> customerInfos = new ObservableCollection<CustomerInformation>();

foreach (var singleItem in obj as ObservableCollection<CustomerInformation>)

{

customerInfos.Add(singleItem as CustomerInformation);

}

foreach (CustomerInformation singleCutomer in customerInfos)

{

cvDatabase.CustomerInformations.DeleteObject(cvDatabase.CustomerInformations.First(x => x.UserID.Equals(singleCutomer.UserID)));

(this.UserMaintenanceCutomerGridView.ItemsSource as ObservableCollection<CustomerInformation>).Remove(singleCutomer);

}

cvDatabase.SaveChanges();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 5], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("User Maintenance All Delete", ErrorException);

DXMessageBox.Show(ErrorException.Message);

}

finally

{

Mouse.OverrideCursor = null;

this.UserMaintenanceDeleteAll.IsEnabled = true;

}

}

#endregion

#region Panel Team Maintenance

public ICommand TeamMainSearchCommand

{

get { return new ReplayCommand(new Action<object>(this.teamMaintenanceSearch)); }

}

public ICommand TeamMainDeleteCommand

{

get { return new ReplayCommand(new Action<object>(this.teamMaintenanceDelete)); }

}

public ICommand TeamMainDeleteAllCommand

{

get { return new ReplayCommand(new Action<object>(this.teamMaintenanceDeleteAll)); }

}

/// <summary>

/// Search User`s

/// </summary>

/// <param name="obj">

/// obj[0] = Name

/// obj[1] = FirstDate

/// obj[2] = Second Date

/// obj[3] = Third Date

/// obj[4] = Minutes

/// </param>

private void teamMaintenanceSearch(object obj)

{

this.temMainSearch.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is ArrayList)

{

ArrayList dataList = obj as ArrayList;

IQueryable<Team> searchQuery = null;

Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities();

switch (this.Option)

{

case "ByName":

string teamName = (string)dataList[0];

searchQuery = cvDatabase.Teams.Where(x => x.Name == teamName);

break;

case "ByDate":

switch (this.SubOption)

{

case "ByDate":

DateTime joinDateEqual = (DateTime)dataList[1];

searchQuery = cvDatabase.Teams.Where(x => x.JoinDate == joinDateEqual);

break;

case "Below":

DateTime joinDateBelow = (DateTime)dataList[1];

searchQuery = cvDatabase.Teams.Where(x => x.JoinDate <= joinDateBelow);

break;

case "TwoDate":

DateTime firstDate = (DateTime)dataList[2];

DateTime seconDate = (DateTime)dataList[3];

searchQuery = cvDatabase.Teams.Where(x => x.JoinDate <= firstDate && x.JoinDate >= seconDate);

break;

default:

break;

}

break;

case "ByAmount":

var minutes = (double)dataList[4];

switch (this.SubOption)

{

case "Below":

searchQuery = cvDatabase.Teams.Where(x => x.TeamAccount.Minutes <= minutes);

break;

case "Equal":

searchQuery = cvDatabase.Teams.Where(x => x.TeamAccount.Minutes == minutes);

break;

}

break;

default:

break;

}

this.teamMainGridView.ItemsSource = new ObservableCollection<Team>(searchQuery);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Team Maintenance Search", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.temMainSearch.IsEnabled = true;

}

}

/// <summary>

/// Delete Selected User

/// </summary>

/// <param name="obj">GridView.SelectedItems</param>

private void teamMaintenanceDelete(object obj)

{

this.teamMainDelete.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<Team> customerInfos = new ObservableCollection<Team>();

foreach (var singleItem in obj as IEnumerable)

{

customerInfos.Add(singleItem as Team);

}

foreach (Team singleCutomer in customerInfos)

{

cvDatabase.Teams.DeleteObject(cvDatabase.Teams.First(x=>x.Name.Equals(singleCutomer.Name)));

(this.teamMainGridView.ItemsSource as ObservableCollection<Team>).Remove(singleCutomer);

}

cvDatabase.SaveChanges();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 5], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Team Maintenance Delete", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.teamMainDelete.IsEnabled = true;

}

}

/// <summary>

/// Delete All User Info

/// </summary>

/// <param name="obj">Gridview.ItemSource</param>

private void teamMaintenanceDeleteAll(object obj)

{

this.teamMainDeleteAll.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

ObservableCollection<Team> customerInfos = new ObservableCollection<Team>();

foreach (var singleItem in obj as ObservableCollection<Team>)

{

customerInfos.Add(singleItem as Team);

}

foreach (Team singleCutomer in obj as ObservableCollection<Team>)

{

cvDatabase.Teams.DeleteObject(cvDatabase.Teams.First(x => x.Name.Equals(singleCutomer.Name)));

(this.teamMainGridView.ItemsSource as ObservableCollection<Team>).Remove(singleCutomer);

}

cvDatabase.SaveChanges();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 5], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Team Maintenance DeleteAll", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.teamMainDeleteAll.IsEnabled = true;

}

}

#endregion

#region Panel Send Mail

public ICommand EmailSendCommand

{

get { return new ReplayCommand(new Action<object>(this.emailSend)); }

}

public ICommand EmailSendingCancelCommand

{

get { return new ReplayCommand(new Action<object>(this.emailSendingCancel)); }

}

/// <summary>

/// obj[0] = Subject

/// obj[1] = body

/// obj[2] = Name

/// </summary>

/// <param name="obj"></param>

private void emailSend(object obj)

{

this.sendMailSend.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

this.sendMailWorker.RunWorkerAsync(obj);

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Email Sending", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

this.sendMailSend.IsEnabled = true;

}

finally

{

Mouse.OverrideCursor = null;

this.sendMailCancel.IsEnabled = true;

}

}

private void emailSendingCancel(object obj)

{

this.sendMailCancel.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

this.sendMailWorker.CancelAsync();

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Email Sending Cancel", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

this.sendMailCancel.IsEnabled = true;

}

finally

{

Mouse.OverrideCursor = null;

this.sendMailSend.IsEnabled = true;

}

}

private void sendMailWorker\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

if (e.Error != null)

{

DXMessageBox.Show(e.Error.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (e.Cancelled)

{

DXMessageBox.Show("Sending Email Canceled", CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Warning);

}

else

{

DXMessageBox.Show("Email`s are send successfully", CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

private void sendMailWorker\_ProgressChanged(object sender, ProgressChangedEventArgs e)

{

this.sendMailPersentage.Text = string.Format("{0}%", e.ProgressPercentage);

this.sendMailProgress.Value = e.ProgressPercentage;

}

private void sendMailWorker\_DoWork(object sender, DoWorkEventArgs e)

{

try

{

ArrayList dataList = e.Argument as ArrayList;

List<string> toAddress = new List<string>();

using (Cafeteria\_Vernier\_dbEntities cvDatbase = new Cafeteria\_Vernier\_dbEntities())

{

switch (this.Option)

{

case "OneByOne":

string customerName = dataList[2].ToString();

toAddress = cvDatbase.CustomerInformations.Where(x => x.UserID.Equals(customerName) && x.Email != null).Select(x => x.Email).ToList();

break;

case "EveryOne":

toAddress = cvDatbase.CustomerInformations.Where(x => x.Email != null).Select(x => x.Email).ToList();

break;

default:

break;

}

}

this.Dispatcher.Invoke(new Action(() => this.sendMailProgress.Maximum = toAddress.Count()));

ProcestaSendMail sendEmails = new ProcestaSendMail();

string userEmail = Properties.Settings.Default.settingEmail;

string userPassword = Properties.Settings.Default.settingEmailPassword;

string emailSubject = dataList[0].ToString();

string emailBody = dataList[1].ToString();

for (int i = 0; i < toAddress.Count(); i++)

{

if (this.sendMailWorker.CancellationPending)

{

e.Cancel = true;

return;

}

else

{

sendEmails.SendingMail(userEmail, userPassword, "smtp.gmail.com", 587, toAddress[i], emailSubject, emailBody);

this.sendMailWorker.ReportProgress(i);

}

}

}

catch

{

throw;

}

}

#endregion

#region panel New Employ

public ICommand NewEmployeeCommand

{

get { return new ReplayCommand(new Action<object>(this.newEmployeeClick));}

}

public ICommand UpdateEmployeeCommand

{

get { return new ReplayCommand(new Action<object>(this.updateEmployeeClick)); }

}

public ICommand DeleteEmployeeCommand

{

get { return new ReplayCommand(new Action<object>(this.deleteEmployeeClick)); }

}

public ICommand ReportEmployeeCommand

{

get { return new ReplayCommand(new Action<object>(this.reportEmployeeClick)); }

}

public ICommand BrowseEmployeeCommand

{

get { return new ReplayCommand(new Action<object>(this.browseEmployeeClick)); }

}

public ICommand WebCamEmployeeCommand

{

get { return new ReplayCommand(new Action<object>(this.webCamEmployeeClick)); }

}

/// <summary>

/// Add new Employ Information

/// </summary>

/// <param name="obj">In this Case Parameter is not need.</param>

private void newEmployeeClick(object obj)

{

this.EmployeeNew.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

ViewEmployPermissions defaultPermission = new ViewEmployPermissions();

ModelEmployee newEmployee = new ModelEmployee();

newEmployee.Permissions = new ObservableCollection<ModelEmployPermissions>(defaultPermission.Where(x=>x.SupperTip!=null).OrderBy(x=>x.Priority));

(this.EmployeeGridView.ItemsSource as ObservableCollection<ModelEmployee>).Add(newEmployee);

this.EmployeeGridView.Rebind();

this.EmployeeGridView.SelectedItem = this.EmployeeGridView.Items[this.EmployeeGridView.Items.IndexOf(newEmployee)];

}

catch (Exception errorException)

{

LogFileWriter.ErrorToLog("New Employee Click",errorException);

DXMessageBox.Show(errorException.Message, CvVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.EmployeeNew.IsEnabled = true;

}

}

/// <summary>

/// Update or Insert Employee Information

/// </summary>

/// <param name="obj">RadGridView selected item</param>

private void updateEmployeeClick(object obj)

{

this.EmployeeUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj!=null)

{

ModelEmployee selectedEmployee = obj as ModelEmployee;

using (Cafeteria\_Vernier\_dbEntities CVDatabase = new Cafeteria\_Vernier\_dbEntities())

{

var employeeExist = CVDatabase.Employees.FirstOrDefault(x => x.EmployeeID.Equals(selectedEmployee.Name));

if (employeeExist!=null)

{

if (this.LoginEmployee.Permissions.AsParallel().FirstOrDefault(x => x.Item.Equals("EmployeeEdit")) != null)

{

Mouse.OverrideCursor = null;

if (DXMessageBox.Show(CvVariables.ERROR\_MESSAGES[1, 8], CvVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

Mouse.OverrideCursor = Cursors.Wait;

var existinfPermissions = CVDatabase.EmployeePermissions.Where(x => x.EmployeeID.Equals(selectedEmployee.Name));

foreach (var permission in existinfPermissions)

{

CVDatabase.EmployeePermissions.DeleteObject(permission);

}

employeeExist.Address = selectedEmployee.Address;

employeeExist.Password = String.IsNullOrEmpty(selectedEmployee.Password) ? employeeExist.Password : selectedEmployee.Password;

employeeExist.Phone = selectedEmployee.PhoneNmber;

employeeExist.UserImage = selectedEmployee.UserImage;

foreach (var permission in selectedEmployee.Permissions.Where(x=>x.Permission.Equals(true)))

{

employeeExist.EmployeePermissions.Add(new EmployeePermission { AutoInc=default(long), EmployeeID=selectedEmployee.Name, Privilege=permission.Item, SettingPrivilage = permission.Setting });

}

CVDatabase.SaveChanges();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 2], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

return;

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 1], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Hand);

}

}

else

{

Employee newEmployee = new Employee { Address=selectedEmployee.Address, EmployeeID=selectedEmployee.Name, Password=selectedEmployee.Password, Phone=selectedEmployee.PhoneNmber, UserImage=selectedEmployee.UserImage };

foreach (var permission in selectedEmployee.Permissions.Where(x=>x.Permission.Equals(true)))

{

newEmployee.EmployeePermissions.Add(new EmployeePermission { AutoInc=default(long), EmployeeID=selectedEmployee.Name, Privilege=permission.Item, SettingPrivilage=permission.Setting });

}

CVDatabase.Employees.AddObject(newEmployee);

CVDatabase.SaveChanges();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 3], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0,4],CvVariables.ERROR\_MESSAGES[0,0],MessageBoxButton.OK,MessageBoxImage.Hand);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Update Or Insert Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message,CvVariables.ERROR\_MESSAGES[0,0],MessageBoxButton.OK,MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.EmployeeUpdate.IsEnabled = true;

}

}

/// <summary>

/// Delete Employee Information

/// </summary>

/// <param name="obj">RadGridView selected item</param>

private void deleteEmployeeClick(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj!=null)

{

if (DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0,6],CvVariables.SOFTWARE\_NAME,MessageBoxButton.YesNo,MessageBoxImage.Question)==MessageBoxResult.Yes)

{

this.EmployeeDelete.IsEnabled = false;

ModelEmployee deleteEmployee = obj as ModelEmployee;

using (Cafeteria\_Vernier\_dbEntities CVDatabase = new Cafeteria\_Vernier\_dbEntities())

{

CVDatabase.Employees.DeleteObject(CVDatabase.Employees.First(x => x.EmployeeID.Equals(deleteEmployee.Name)));

CVDatabase.SaveChanges();

(this.EmployeeGridView.ItemsSource as ObservableCollection<ModelEmployee>).Remove(deleteEmployee);

this.EmployeeGridView.Rebind();

this.selectGridViewFirstItem(this.EmployeeGridView);

}

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 5], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

return;

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0,4],CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Exclamation);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Employee Delete Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message,CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.EmployeeDelete.IsEnabled = true;

}

}

/// <summary>

/// Make a Report based on all employee information

/// </summary>

/// <param name="obj">RadGridView ItemSource </param>

private void reportEmployeeClick(object obj)

{

this.EmployeeReport.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj !=null)

{

ModelEmployee selectedEmployeeReport = obj as ModelEmployee;

ReportEmployee reportSelectedEmployee = new ReportEmployee();

reportSelectedEmployee.EmployeePermissions.DataSource = new ObservableCollection<ModelEmployPermissions>(from selectedEmployeePermission in selectedEmployeeReport.Permissions.AsParallel() select new ModelEmployPermissions { Item = selectedEmployeePermission.SupperTip, ScreenTip = selectedEmployeePermission.Permission ? "Yes" : "No" });

reportSelectedEmployee.EmployeeInfo.DataSource = selectedEmployeeReport;

PrintHelper.ShowPrintPreview(this, reportSelectedEmployee);

Mouse.OverrideCursor = null;

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 4], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Exclamation);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Employee Report Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.EmployeeReport.IsEnabled = true;

}

}

/// <summary>

/// Set a Image to Employee profile from file.

/// </summary>

/// <param name="obj">RadGridView Selected Item</param>

private void browseEmployeeClick(object obj)

{

this.EmployeeBrowse.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj!=null)

{

(obj as ModelEmployee).UserImage = this.selectImageFromFile();

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 4], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Exclamation);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Employee Browse Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.EmployeeBrowse.IsEnabled = true;

}

}

/// <summary>

/// Set a Image to Employee profile from a Web Cam.

/// </summary>

/// <param name="obj">RadGridView Selected Item</param>

private void webCamEmployeeClick(object obj)

{

this.EmployeeWebCam.IsEnabled = false;

try

{

if (obj != null)

{

byte[] imageInBytes = null;

imageInBytes = WebCamWindow.CaptureImage();

Mouse.OverrideCursor = Cursors.Wait;

if (imageInBytes!=null)

{

(obj as ModelEmployee).UserImage=imageInBytes;

}

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 4], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Exclamation);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Employee WebCam Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.EmployeeWebCam.IsEnabled = true;

}

}

/// <summary>

/// Permission DataGrid CheckAll

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void EmployeePermissionChecked(object sender, System.Windows.RoutedEventArgs e)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

Parallel.ForEach((this.EmployeePermission.ItemsSource as ObservableCollection<ModelEmployPermissions>), currentPermission =>

{

currentPermission.Permission = true;

});

this.EmployeePermission.Rebind();

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Employee CheckAll Checked", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

}

}

private void EmployeePermissionUnchecked(object sender, System.Windows.RoutedEventArgs e)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

Parallel.ForEach((this.EmployeePermission.ItemsSource as ObservableCollection<ModelEmployPermissions>), currentPermission =>

{

currentPermission.Permission = false;

});

this.EmployeePermission.Rebind();

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Employee CheckAll Checked", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

}

}

#endregion

#region Panel Database Backup And Restore

public ICommand BackupLocationCommand

{

get { return new ReplayCommand(new Action<object>(this.browseBackupLocation\_Click)); }

}

public ICommand DatabaseBackupCommand

{

get { return new ReplayCommand(new Action<object>(this.databaseBackup\_Click)); }

}

public ICommand RestoreLocationCommand

{

get { return new ReplayCommand(new Action<object>(this.browseRestoreLocation\_Click)); }

}

public ICommand DatabaseRestore

{

get { return new ReplayCommand(new Action<object>(this.databaseRestore\_Click)); }

}

private void browseBackupLocation\_Click(object obj)

{

this.dbbackupBrowse.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

System.Windows.Forms.FolderBrowserDialog backupPathDialog = new System.Windows.Forms.FolderBrowserDialog();

backupPathDialog.Description = "Please Select a folder for backup file";

backupPathDialog.ShowNewFolderButton = true;

if (backupPathDialog.ShowDialog().Equals(System.Windows.Forms.DialogResult.OK))

{

this.dbbackupFolderPath.Text = backupPathDialog.SelectedPath;

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Database backup folder browse Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.dbbackupBrowse.IsEnabled = true;

}

}

private void databaseBackup\_Click(object obj)

{

this.dbbackupBackup.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

this.dbBackupWorker.RunWorkerAsync(obj);

}

private void browseRestoreLocation\_Click(object obj)

{

this.dbrestoreBrowse.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

System.Windows.Forms.OpenFileDialog restorePathDialog = new System.Windows.Forms.OpenFileDialog();

restorePathDialog.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

restorePathDialog.Filter = "SQL Backup file (\*.bak)|\*.bak";

if (restorePathDialog.ShowDialog().Equals(System.Windows.Forms.DialogResult.OK))

{

this.dbrestoreFilePath.Text = restorePathDialog.FileName;

FileInfo selectedFileInfo = new FileInfo(restorePathDialog.FileName);

this.dbRestoreDate.Text = selectedFileInfo.CreationTime.ToString();

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Database restore file browse Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.dbrestoreBrowse.IsEnabled = true;

}

}

private void databaseRestore\_Click(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

this.dbrestoreRestore.IsEnabled = false;

this.dbRestoreWorker.RunWorkerAsync(obj);

}

private void dbRestoreWorker\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

Mouse.OverrideCursor = null;

if (e.Error != null)

{

DXMessageBox.Show(e.Error.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 8], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

this.dbrestoreRestore.IsEnabled = true;

}

void dbRestoreWorker\_ProgressChanged(object sender, ProgressChangedEventArgs e)

{

if (e.ProgressPercentage==0)

{

this.dbRestoreProgress.IsIndeterminate = true;

}

else if (e.ProgressPercentage==1)

{

this.dbRestoreProgress.IsIndeterminate = false;

}

else

{

this.dbRestoreProgress.Value = e.ProgressPercentage;

}

}

void dbRestoreWorker\_DoWork(object sender, DoWorkEventArgs e)

{

try

{

using (SqlConnection restoreConnection = new SqlConnection(@"Data Source=.\SQLEXPRESS;Initial Catalog=master;Integrated Security=True"))

{

restoreConnection.Open();

this.dbRestoreWorker.ReportProgress(10);

System.Threading.Thread.Sleep(200);

SqlCommand UseMasterCommand = new SqlCommand("USE master", restoreConnection);

UseMasterCommand.ExecuteNonQuery();

string Alter1 = string.Format(@"ALTER DATABASE [{0}] SET Single\_User WITH Rollback Immediate", CvVariables.Catalog);

SqlCommand Alter1Cmd = new SqlCommand(Alter1, restoreConnection);

Alter1Cmd.ExecuteNonQuery();

Alter1Cmd.Dispose();

this.dbRestoreWorker.ReportProgress(20);

System.Threading.Thread.Sleep(200);

this.dbRestoreWorker.ReportProgress(0);

string Restore = string.Format(@"RESTORE DATABASE [{0}] FROM DISK = N'{1}' WITH FILE = 1, NOUNLOAD, STATS = 10,MOVE '{0}' TO " + @"'{2}\{0}.mdf',MOVE '{0}\_log' TO '{2}\{0}\_log.ldf'", CvVariables.Catalog, e.Argument.ToString(), Properties.Settings.Default.SqlDataFolder);

SqlCommand RestoreCmd = new SqlCommand(Restore, restoreConnection);

RestoreCmd.ExecuteNonQuery();

RestoreCmd.Dispose();

this.dbRestoreWorker.ReportProgress(1);

string Alter2 = string.Format(@"ALTER DATABASE [{0}] SET Multi\_User", CvVariables.Catalog);

SqlCommand Alter2Cmd = new SqlCommand(Alter2, restoreConnection);

Alter2Cmd.ExecuteNonQuery();

Alter2Cmd.Dispose();

System.Threading.Thread.Sleep(200);

this.dbRestoreWorker.ReportProgress(30);

}

}

catch (Exception)

{

throw;

}

}

void dbBackupWorker\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

Mouse.OverrideCursor = null;

if (e.Error!=null)

{

DXMessageBox.Show(e.Error.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0,7], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

this.dbbackupBackup.IsEnabled = true;

}

void dbBackupWorker\_ProgressChanged(object sender, ProgressChangedEventArgs e)

{

if (e.ProgressPercentage==0)

{

this.dbBackupProgress.IsIndeterminate = true;

}

else if (e.ProgressPercentage==1)

{

this.dbBackupProgress.IsIndeterminate = false;

}

else

{

this.dbBackupProgress.Value = e.ProgressPercentage;

}

}

void dbBackupWorker\_DoWork(object sender, DoWorkEventArgs e)

{

try

{

using (SqlConnection backupConnection = new SqlConnection(Properties.Settings.Default.Cafeteria\_Vernier\_dbConnectionString))

{

backupConnection.Open();

dbBackupWorker.ReportProgress(10);

System.Threading.Thread.Sleep(200);

ServerConnection sc = new ServerConnection(backupConnection);

Server databaseServer = new Server(sc);

Backup databaseBackup = new Backup();

databaseBackup.Action = BackupActionType.Database;

dbBackupWorker.ReportProgress(20);

System.Threading.Thread.Sleep(200);

databaseBackup.Database = backupConnection.Database.ToString();

databaseBackup.Devices.Add(new BackupDeviceItem(System.IO.Path.Combine(e.Argument.ToString(), string.Format("{0}.bak", DateTime.Now.ToString("ddMMyyyyhhsstt"))), DeviceType.File));

databaseBackup.LogTruncation = BackupTruncateLogType.Truncate;

dbBackupWorker.ReportProgress(0);

databaseBackup.SqlBackup(databaseServer);

dbBackupWorker.ReportProgress(1);

System.Threading.Thread.Sleep(200);

sc.ForceDisconnected();

dbBackupWorker.ReportProgress(30);

}

}

catch (Exception)

{

throw;

}

}

#endregion

#region User Status Reset

public ICommand UserStatusResetCommand

{

get { return new ReplayCommand(new Action<object>(this.userStatusReset\_Click)); }

}

public ICommand TeamStatusResetCommand

{

get { return new ReplayCommand(new Action<object>(this.teamStatusReset\_Click)); }

}

private void userStatusReset\_Click(object obj)

{

this.userResButtonUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

CustomerAccount userStatus = obj as CustomerAccount;

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

cvDatabase.CustomerAccounts.First(x => x.UserID.Equals(userStatus.UserID)).Status = false;

cvDatabase.SaveChanges();

this.mainMenuClick("CustomerStatusReset");

Mouse.OverrideCursor = null;

DXMessageBox.Show("Customer Status reset successfully", CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("User Reset Update Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.userResButtonUpdate.IsEnabled = true;

}

}

private void teamStatusReset\_Click(object obj)

{

this.TeamResButtonUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

TeamAccount userStatus = obj as TeamAccount;

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

cvDatabase.TeamAccounts.First(x => x.Name.Equals(userStatus.Name)).Status = false;

cvDatabase.SaveChanges();

this.mainMenuClick("CustomerStatusReset");

Mouse.OverrideCursor = null;

DXMessageBox.Show("Team Status reset successfully", CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Team Reset Update Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.TeamResButtonUpdate.IsEnabled = true;

}

}

#endregion

#region Panel Setting

public ICommand SettingMenuCommand

{

get { return new ReplayCommand(new Action<object>(this.settingMenu\_Click)); }

}

private void settingMenu\_Click(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

IEnumerable<Grid> settingPanels = this.PanelSetting.Children.OfType<Grid>().Where(x => x.Visibility == Visibility.Visible);

foreach (Grid panel in settingPanels)

{

panel.Visibility = Visibility.Hidden;

}

switch (obj.ToString())

{

case "RateSetting":

this.SettingPanelRateSetup.Visibility = Visibility.Visible;

break;

case "ChangePassword":

this.settingChangeOldPassword.Password = string.Empty;

this.settingChangeNewPassword.Password = string.Empty;

this.settingChnageConPassword.Password = string.Empty;

this.settingPanelChangePassword.Visibility = Visibility.Visible;

break;

case "ScreenCapture":

this.SettingPanelScreenCapture.Visibility = Visibility.Visible;

break;

case "EMailSetting":

this.SettingPanelEmailSetting.Visibility = Visibility.Visible;

break;

case "GeneralSetting":

this.settingGeneralCheckSystemStart.IsChecked = this.CheckStartupValue();

this.SettingPanelGeneral.Visibility = Visibility.Visible;

break;

default:

break;

}

Mouse.OverrideCursor = null;

}

#region Panel Minutes Setting

public ICommand RateSettingNewCommand

{

get { return new ReplayCommand(new Action<object>(this.rateSettingNew\_Click)); }

}

public ICommand RateSettingUpdateCommand

{

get { return new ReplayCommand(new Action<object>(this.rateSettingUpdate\_Click)); }

}

public ICommand RateSettingDeleteCommand

{

get { return new ReplayCommand(new Action<object>(this.rateSettingDelete\_Click)); }

}

private void rateSettingNew\_Click(object obj)

{

this.RateSettingNew.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

ModelBillConfig newRateInfo = new ModelBillConfig();

(this.RateSettingsGrid.ItemsSource as ObservableCollection<ModelBillConfig>).Add(newRateInfo);

this.RateSettingsGrid.SelectedItem = this.RateSettingsGrid.Items[this.RateSettingsGrid.Items.IndexOf(newRateInfo)];

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Rate Setup New Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.RateSettingNew.IsEnabled = true;

}

}

/// <summary>

/// Update Rate Configuration File

/// </summary>

/// <param name="obj">Grid Panel DataContext</param>

private void rateSettingUpdate\_Click(object obj)

{

this.RateSettingUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

this.updateRateXml(this.BillConfigInfo);

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0,2], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Rate Setup Update Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.RateSettingUpdate.IsEnabled = true;

}

}

/// <summary>

/// Delete Rate

/// </summary>

/// <param name="obj">

/// ComboBox Selected Item (Minutes)

/// </param>

private void rateSettingDelete\_Click(object obj)

{

this.RateSettingDelete.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

this.BillConfigInfo.Remove(obj as ModelBillConfig);

this.updateRateXml(this.BillConfigInfo);

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 5], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Rate Setup Delete Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.RateSettingDelete.IsEnabled = true;

}

}

#endregion

#region Panel Change Password

public ICommand PasswordChangeCommand

{

get{return new ReplayCommand(new Action<object>(this.passwordChange\_Click));}

}

private void passwordChange\_Click(object obj)

{

this.settingChageUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

var employeeInfo = cvDatabase.Employees.Where(x => x.EmployeeID.Equals(LoginEmployee.Name) && x.Password.Equals(LoginEmployee.Password)).First();

employeeInfo.Password = obj.ToString();

cvDatabase.SaveChanges();

this.LoginEmployee.Password = obj.ToString();

}

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 9], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Password change Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.settingChageUpdate.IsEnabled = true;

}

}

#endregion

#region Panel screen Capture

public ICommand ScrSnapFolderBrowseCommand

{

get { return new ReplayCommand(new Action<object>(this.scrSnapFolderBrowse\_Click)); }

}

public ICommand ScrSnapStratCommand

{

get { return new ReplayCommand( new Action<object>(this.scrSnapStart\_Click));}

}

public ICommand ScrSnapStopCommand

{

get { return new ReplayCommand(new Action<object>(this.scrSnapStop\_Click)); }

}

private void scrSnapFolderBrowse\_Click(object obj)

{

this.settingScrCapBrowse.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

using (System.Windows.Forms.FolderBrowserDialog folderBrowse = new System.Windows.Forms.FolderBrowserDialog())

{

if (folderBrowse.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

if (folderBrowse.SelectedPath.StartsWith("C:\\"))

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 10], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Stop);

return;

}

Properties.Settings.Default.schreenCapturePath = folderBrowse.SelectedPath;

Properties.Settings.Default.Save();

}

}

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Screen snapshot browse Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.settingScrCapBrowse.IsEnabled = true;

}

}

private void scrSnapStart\_Click(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

Properties.Settings.Default.CuptureTime = (TimeSpan)obj;

Properties.Settings.Default.IsCupture = true;

Properties.Settings.Default.Save();

ScreenShort.start();

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Screen snapshot start Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

}

}

private void scrSnapStop\_Click(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

Properties.Settings.Default.IsCupture = false;

Properties.Settings.Default.Save();

ScreenShort.Stop();

}

catch (Exception ErrorException)

{

LogFileWriter.ErrorToLog("Screen snapshot stop Click", ErrorException);

Mouse.OverrideCursor = null;

DXMessageBox.Show(ErrorException.Message, CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

}

}

#endregion

#region Panel General

/// <summary>

/// Start with System Start CheckBox Checked

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void settingGeneralCheckSystemStartChecked(object sender, System.Windows.RoutedEventArgs e)

{

Registry.CurrentUser.OpenSubKey(@"Software\Microsoft\Windows\CurrentVersion\Run", true).SetValue("CvServer", System.Reflection.Assembly.GetEntryAssembly().Location, RegistryValueKind.String);

}

/// <summary>

/// Start with System Start CheckBox UnChecked

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void settingGeneralCheckSystemStartUnChecked(object sender, System.Windows.RoutedEventArgs e)

{

if (this.CheckStartupValue())

{

using (RegistryKey regKey = Registry.CurrentUser.OpenSubKey(@"Software\Microsoft\Windows\CurrentVersion\Run", true))

{

if (regKey != null)

{

regKey.DeleteValue("CvServer");

}

}

}

}

#endregion

#endregion

#region Private Methods

private void selectGridViewFirstItem(RadGridView radGridView)

{

if (radGridView.Items.Count>0)

{

radGridView.SelectedItem = radGridView.Items[0];

}

}

private void selectListBoxFirstItem(ListBox listBox)

{

if (listBox.Items.Count>0)

{

listBox.SelectedIndex = 0;

}

}

private byte[] selectImageFromFile()

{

byte[] imageInbytes = null;

using (System.Windows.Forms.OpenFileDialog imageOpenBox = new System.Windows.Forms.OpenFileDialog())

{

imageOpenBox.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);

imageOpenBox.Filter = "JPGE (\*.jpg)|\*.jpg|PNG (\*.png)|\*.png|BMP (\*.bmp)|\*.bmp|All (\*.\*)|\*.\*";

imageOpenBox.FilterIndex = 0;

imageOpenBox.RestoreDirectory = true;

if (imageOpenBox.ShowDialog().Equals(System.Windows.Forms.DialogResult.OK))

{

imageInbytes = new MiraculousMethods().imageToByteArray(imageOpenBox.FileName);

}

}

return imageInbytes;

}

private void getValidationError(params DependencyObject[] dp)

{

foreach (DependencyObject depenency in dp)

{

foreach (ValidationError errors in Validation.GetErrors(depenency))

{

throw new Exception(errors.ErrorContent.ToString());

}

}

}

private void hidePanels()

{

var visiablePanels = PanelRoot.Children.OfType<Grid>().Where(x => x.Visibility.Equals(Visibility.Visible));

foreach (var panel in visiablePanels)

{

panel.Visibility = Visibility.Hidden;

}

}

private ObservableCollection<ModelCommonUse> customerInfo()

{

ObservableCollection<ModelCommonUse> customerShortInfo;

using (Cafeteria\_Vernier\_dbEntities cvDatabse = new Cafeteria\_Vernier\_dbEntities())

{

customerShortInfo = new ObservableCollection<ModelCommonUse>(cvDatabse.CustomerInformations.Select(x =>

new ModelCommonUse

{

UserName = x.UserID,

Image = x.Logo

}));

}

return customerShortInfo;

}

private void updateRateXml(ObservableCollection<ModelBillConfig> bilConfigs)

{

string billXmlPath = System.IO.Path.Combine(new MiraculousMethods().GetTempFolder(CvVariables.SOFTWARE\_NAME), CvVariables.MUNITIES\_FILE);

if (!File.Exists(billXmlPath))

{

XDocument newRateConfig = new XDocument(new XDeclaration("1.0", "utf-8", "yes"), new XElement("Rates"));

newRateConfig.Save(billXmlPath);

}

XElement loadRateConfig = XElement.Load(billXmlPath);

loadRateConfig.Elements("Rate").Remove();

foreach (ModelBillConfig rateInfo in bilConfigs)

{

loadRateConfig.Add(new XElement("Rate", new XElement("Minutes", rateInfo.Minutes), new XElement("Bill", rateInfo.Amount)));

}

loadRateConfig.Save(billXmlPath);

}

#region Methods

/// <summary>

/// On Closing

/// </summary>

/// <param name="e"></param>

protected override void OnClosing(System.ComponentModel.CancelEventArgs e)

{

Properties.Settings.Default.Save();

MessageBoxResult messBoxResult = DXMessageBox.Show(CvVariables.ERROR\_MESSAGES[0, 4], CvVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.YesNo, MessageBoxImage.Question);

if (messBoxResult.Equals(MessageBoxResult.Yes))

{

ScreenShort.Stop();

base.OnClosing(e);

Application.Current.Shutdown();

}

else

{

e.Cancel = true;

}

}

/// <summary>

/// Check CvServer Value Exit or Not

/// </summary>

/// <returns></returns>

public bool CheckStartupValue()

{

using (RegistryKey regKey = Registry.CurrentUser.OpenSubKey(@"Software\Microsoft\Windows\CurrentVersion\Run", true))

{

if (regKey != null)

{

object keyValue = regKey.GetValue("CvServer");

if (keyValue!=null)

{

return true;

}

else

{

return false;

}

}

else

{

return false;

}

}

}

#endregion

#endregion

#region Propery Change

public event PropertyChangedEventHandler PropertyChanged;

private void OnPropertyChanged(string PropertyName)

{

if (PropertyChanged != null)

{

this.PropertyChanged(this, new PropertyChangedEventArgs(PropertyName));

}

}

#endregion

#region ICommand Class

public class ReplayCommand : ICommand

{

private Action<object> \_action;

public ReplayCommand(Action<object> action)

{

this.\_action = action;

}

public bool CanExecute(object parameter)

{

return true;

}

#pragma warning disable 67

public event EventHandler CanExecuteChanged;

#pragma warning restore 67

public void Execute(object parameter)

{

try

{

if (parameter != null)

{

this.\_action(parameter);

}

else

{

DXMessageBox.Show(CvVariables.ERROR\_MESSAGESS[0, 0], CvVariables.SOFTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Hand);

}

}

catch (Exception ex)

{

LogFileWriter.ErrorToLog("Replay Command", ex);

DXMessageBox.Show(ex.Message,CvVariables.SOFTWARE\_NAME,MessageBoxButton.OK,MessageBoxImage.Error);

}

}

}

#endregion

}

}

Class: StratWindow

namespace Procesta.CvServer

{

/// <summary>

/// Interaction logic for StratWindow.xaml

/// </summary>

public partial class StratWindow : Window,INotifyPropertyChanged

{

private BackgroundWorker startBackgroundWorker = new BackgroundWorker();

private MainWindow serverWindow;

/// <summary>

/// Window instillation

/// </summary>

public StratWindow()

{

this.InitializeComponent();

startBackgroundWorker.WorkerReportsProgress = true;

startBackgroundWorker.DoWork += new DoWorkEventHandler(startBackgroundWorker\_DoWork);

startBackgroundWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(startBackgroundWorker\_RunWorkerCompleted);

startBackgroundWorker.RunWorkerAsync();

}

/// <summary>

///BackGroundWorker Work Complete

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

void startBackgroundWorker\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

if (!e.Cancelled && e.Error==null)

{

// new MainWindow().Show();

if (serverWindow!=null)

{

serverWindow.Show();

}

this.Close();

}

else

{

/// Code here to write log file

Application.Current.Shutdown();

}

}

/// <summary>

/// BackGroundWorker Do Work

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

void startBackgroundWorker\_DoWork(object sender, DoWorkEventArgs e)

{

this.UpperText = "Connecting";

this.LowerText = "Database";

Thread.Sleep(1000);

using (SqlConnection testConnection = new SqlConnection(Properties.Settings.Default.Cafeteria\_Vernier\_dbConnectionString))

{

try

{

testConnection.Open();

}

catch

{

this.UpperText = "Connection";

this.LowerText = "Fault";

Thread.Sleep(1000);

new InstallWindow().Show();

this.Close();

return;

}

}

this.UpperText = "Connecting";

this.LowerText = "Service";

Thread.Sleep(2000);

//Check connection Named pipe service

this.UpperText = "Retrieving";

this.LowerText = "Necessary Information";

Thread.Sleep(2000);

MiraculousMethods miraculousMethod = new MiraculousMethods();

miraculousMethod.CheckCashDate();

this.Dispatcher.BeginInvoke(new Action(()=>

{

serverWindow = new MainWindow();

serverWindow.BillConfigInfo = miraculousMethod.LoadBillConfig();

}),DispatcherPriority.Normal);

//

//

//miraculousMethod.MinimumRequirement();

}

/// <summary>

/// Start WCF Server at 9078 Port.

/// It run another thread

/// </summary>

private static void startWcfServer()

{

try

{

//ServiceHost duplex = new ServiceHost(typeof(ConnectFromServer));

//NetTcpBinding tcpBinding = new NetTcpBinding();

//tcpBinding.Security.Mode = SecurityMode.None;

//duplex.AddServiceEndpoint(typeof(IServerWithCallback), tcpBinding, string.Format("net.tcp://{0}:9078/DataService", Properties.Settings.Default.ServerIpAddress));

//duplex.Open();

}

catch { }

}

#region Animation Text Property

public string UpperText

{

get { return this.\_UpperText; }

set

{

this.\_UpperText = value;

this.onPropertyChange("UpperText");

}

}

public string LowerText

{

get { return this.\_LowerText; }

set

{

this.\_LowerText = value;

this.onPropertyChange("LowerText");

}

}

private string \_UpperText;

private string \_LowerText;

#endregion

#region Property Chnage

public event PropertyChangedEventHandler PropertyChanged;

private void onPropertyChange(string propertName)

{

if (this.PropertyChanged!=null)

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertName));

}

}

#endregion

}

}

Service Code

Class: ClientNotificationService

[ServiceBehavior(InstanceContextMode=InstanceContextMode.Single, IncludeExceptionDetailInFaults=true)]

public class ClientNotificationService :IClientNotification

{

ObservableCollection<CommandData> ovcCommands = new ObservableCollection<CommandData>();

public void setCommand(CommandData command)

{

try

{

ovcCommands.Add(command);

}

catch

{

throw new FaultException("Unable to send command");

}

}

public ObservableCollection<CommandData> GetCommands(string counterNumber)

{

return new ObservableCollection<CommandData>(ovcCommands.Where(x => x.CounterNumber == counterNumber));

}

public void RemoveCommand(string counterNumber)

{

try

{

this.ovcCommands.Remove(this.ovcCommands.First(x => x.CounterNumber == counterNumber));

}

catch

{

}

}

}

}

**Class: ServerNotificationService**

[ServiceBehavior(InstanceContextMode=InstanceContextMode.Single)]

public class ServerNotificationService :IServerNotification

{

ObservableCollection<CounterInformation> ovcCounterInfo = new ObservableCollection<CounterInformation>();

public void SetCounterInformation(CounterInformation counterInfo)

{

try

{

if (ovcCounterInfo.SingleOrDefault(x => x.CounterNumber == counterInfo.CounterNumber) == null)

{

ovcCounterInfo.Add(counterInfo);

}

}

catch

{

throw new FaultException("Unable to register");

}

}

public ObservableCollection<CounterInformation> GetCounterInformation()

{

return ovcCounterInfo;

}

public void RemoveCounterInformation(CounterInformation counterInfo)

{

try

{

this.ovcCounterInfo.Remove(ovcCounterInfo.FirstOrDefault(x => x.CounterNumber.Equals(counterInfo.CounterNumber)));

}

catch (Exception ex)

{

throw new FaultException(ex.Message);

}

}

}

}

Class: ServerSideServices

namespace Procesta.serverSideService

{

[ServiceBehavior(IncludeExceptionDetailInFaults=true)]

public class ServerSideServices : IServerSideServices

{

private static readonly object signalEntry = new object();

public List<Int64> UserLogin(string username, string password, short counterNumber)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

var loginfo = cvDatabase.CustomerAccounts.FirstOrDefault(x=>x.UserID.Equals(username)

&& x.Password.Equals(password)

&& x.Minutes>1

&& x.Status==false);

if (loginfo!=null)

{

UserLoginHistory newLoginHistory= new UserLoginHistory

{

AutoInc= default(long),

CounterNumber=counterNumber,

StratTime=DateTime.Now,

UserID=username,

};

cvDatabase.AddToUserLoginHistories(newLoginHistory);

loginfo.Status=true;

loginfo.Counternumber=counterNumber;

cvDatabase.SaveChanges();

List<Int64> customerinfo = new List<Int64>();

customerinfo.Add(loginfo.Minutes);

customerinfo.Add(newLoginHistory.AutoInc);

return customerinfo;

}

else

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 0]);

}

}

}

catch (Exception error)

{

throw new FaultException(error.Message);

}

}

public List<Int64> TeamLogin(string username, string password, short counterNumber, string teamName)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

var teamInfo = cvDatabase.TeamAccounts.FirstOrDefault(x => x.Team.Name.Equals(teamName)

&& x.Status == false);

var memberInfo = cvDatabase.TeamMembers.FirstOrDefault(x => x.UserID.Equals(username));

var userInfo = cvDatabase.CustomerAccounts.FirstOrDefault(x => x.UserID.Equals(username) && x.Password.Equals(password));

var isTeamAdmin = cvDatabase.Teams.FirstOrDefault(x => x.Name.Equals(teamName) && x.AdminName.Equals(username));

if (teamInfo!=null && (memberInfo!=null || isTeamAdmin!=null ) && userInfo!=null)

{

UserLoginHistory newUserLoginHistory = new UserLoginHistory

{

AutoInc =default(long),

CounterNumber=counterNumber,

StratTime=DateTime.Now,

TeamName=teamName,

UserID=username

};

cvDatabase.AddToUserLoginHistories(newUserLoginHistory);

teamInfo.Status = true;

var userinfo= cvDatabase.CustomerAccounts.FirstOrDefault(x => x.UserID.Equals(username));

userinfo.Status = false;

userinfo.Counternumber = counterNumber;

cvDatabase.SaveChanges();

List<Int64> teamLoginInfo = new List<Int64>();

teamLoginInfo.Add(teamInfo.Minutes);

teamLoginInfo.Add(newUserLoginHistory.AutoInc);

return teamLoginInfo;

}

else

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 1]);

}

}

}

catch (Exception error)

{

throw new FaultException(error.Message);

}

}

public bool UserMunitieCounter(string username, short counterNumber)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

var userAccount = cvDatabase.CustomerAccounts.FirstOrDefault(x => x.UserID.Equals(username) && x.Minutes > 1);

if (userAccount!=null)

{

userAccount.Minutes--;

userAccount.Status = true;

userAccount.Counternumber = counterNumber;

cvDatabase.SaveChanges();

return true;

}

else

{

return false;

}

}

}

catch (Exception error)

{

throw new FaultException(error.Message);

}

}

public bool TeamMunitieCounter(string teamName,string username, short counterNumber)

{

try

{

lock (signalEntry)

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

var teamAccount = cvDatabase.TeamAccounts.FirstOrDefault(x => x.Name.Equals(teamName) && x.Minutes > 1);

var userAccount = cvDatabase.CustomerAccounts.FirstOrDefault(x => x.UserID.Equals(username));

if (teamAccount != null && userAccount != null)

{

teamAccount.Minutes--;

teamAccount.Status = true;

userAccount.Status = true;

userAccount.Counternumber = counterNumber;

cvDatabase.SaveChanges();

return true;

}

else

{

return false;

}

}

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 4]);

}

}

public Userinformation AccountDetails(string username)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

return (from userinfoTable in cvDatabase.CustomerInformations

where userinfoTable.UserID.Trim().Equals(username)

select new Userinformation

{

Address = userinfoTable.Address,

Date = userinfoTable.JoinDate,

Email = userinfoTable.Email,

Name = userinfoTable.Name,

NationalID = userinfoTable.NationalID,

Phone = userinfoTable.Phone,

UserImage = userinfoTable.Logo

}).SingleOrDefault();

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public bool AccountUpdate(string username, Userinformation updateinfo)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

var customerInfo = cvDatabase.CustomerInformations.First(x => x.UserID.Equals(username));

customerInfo.Address = updateinfo.Address;

customerInfo.Email = updateinfo.Email;

customerInfo.Logo = updateinfo.UserImage;

customerInfo.NationalID = updateinfo.NationalID;

customerInfo.Phone = updateinfo.Phone;

cvDatabase.SaveChanges();

return true;

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0,2]);

}

}

public List<LoginHistory> UserLoginHistoryAll(string username)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

return (from loginHisTable in cvDatabase.UserLoginHistories

where loginHisTable.UserID.Equals(username)

select new LoginHistory

{

CounterNumber = loginHisTable.CounterNumber,

EntryDate = loginHisTable.StratTime,

MinutesUse = (loginHisTable.EndTime.Value.Minute - loginHisTable.StratTime.Minute),

TeamName=loginHisTable.TeamName

}).ToList();

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public List<LoginHistory> UserLoginHistoryDate(string username, DateTime date)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

var data= (from loginHisTable in cvDatabase.UserLoginHistories

where loginHisTable.UserID.Equals(username)

select new LoginHistory

{

CounterNumber = loginHisTable.CounterNumber,

EntryDate = loginHisTable.StratTime,

MinutesUse = loginHisTable.EndTime.Value.Minute - loginHisTable.StratTime.Minute,

TeamName = loginHisTable.TeamName

}).ToList();

return data.Where(x => x.EntryDate.Value.Date == date).ToList();

}

}

catch(Exception ex)

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public List<LoginHistory> UserLoginHistoryTwoDate(string username, DateTime firstDate, DateTime secondDate)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

var data= (from loginHisTable in cvDatabase.UserLoginHistories

where loginHisTable.UserID.Equals(username) && loginHisTable.StratTime.Date <= firstDate && loginHisTable.StratTime.Date >= secondDate

select new LoginHistory

{

CounterNumber = loginHisTable.CounterNumber,

EntryDate = loginHisTable.StratTime,

MinutesUse = (loginHisTable.EndTime.Value.Minute - loginHisTable.StratTime.Minute),

TeamName = loginHisTable.TeamName

}).ToList();

return data.Where(x => x.EntryDate.Value.Date <= firstDate && x.EntryDate.Value.Date >= secondDate).ToList();

}

}

catch(Exception ex)

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public List<UserRechareHistory> UserRechargeHistoryAll(string username)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

return (from rechargeHisTable in cvDatabase.UserRechargeHistories

where rechargeHisTable.UserID.Equals(username)

select new UserRechareHistory

{

Amount = rechargeHisTable.bill,

EmployID=rechargeHisTable.EmployeeID,

EntryDate=rechargeHisTable.DateWithTime,

Munities=rechargeHisTable.Minutes

}).ToList();

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public List<UserRechareHistory> UserRechargeHistoryDate(string username, DateTime date)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

return (from rechargeHisTable in cvDatabase.UserRechargeHistories

where rechargeHisTable.UserID.Equals(username) && rechargeHisTable.DateWithTime == date

select new UserRechareHistory

{

Amount = rechargeHisTable.bill,

EmployID = rechargeHisTable.EmployeeID,

EntryDate = rechargeHisTable.DateWithTime,

Munities = rechargeHisTable.Minutes

}).ToList();

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public List<UserRechareHistory> UserRechargeHistoryTwoDate(string username, DateTime firstDate, DateTime secondDate)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

return (from rechargeHisTable in cvDatabase.UserRechargeHistories

where rechargeHisTable.UserID.Equals(username) && rechargeHisTable.DateWithTime<=firstDate && rechargeHisTable.DateWithTime >=secondDate

select new UserRechareHistory

{

Amount = rechargeHisTable.bill,

EmployID = rechargeHisTable.EmployeeID,

EntryDate = rechargeHisTable.DateWithTime,

Munities = rechargeHisTable.Minutes

}).ToList();

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public List<AllUserAndTeam> GetAllUsers()

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

return (from alluserInfo in cvDatabase.CustomerInformations

select new AllUserAndTeam

{

Name =alluserInfo.UserID.Trim()

}).ToList();

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public bool TeamNameChecker(string teamName)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

if (cvDatabase.Teams.SingleOrDefault(x => x.Name.Trim().Equals(teamName)) == null)

{

return true;

}

else

{

return false;

}

}

}

catch (InvalidOperationException)

{

return false;

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0,2]);

}

}

public bool AddNewTeam(TeamInfo teamInfo, List<AllUserAndTeam> members)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

cvDatabase.AddToTeams(new Team

{

AdminName = teamInfo.TeamAdmin,

JoinDate = teamInfo.EntryDate,

Logo = teamInfo.TeamImage,

Name = teamInfo.TeamD,

TeamAccount = new TeamAccount

{

Minutes = 0,

Name = teamInfo.TeamD,

Status = false,

EntryDate = DateTime.Today

}

});

foreach (AllUserAndTeam teamMember in members)

{

cvDatabase.AddToTeamMembers(new TeamMember

{

AutoInc = default(long),

UserID = teamMember.Name,

Name = teamInfo.TeamD

});

}

cvDatabase.SaveChanges();

}

return true;

}

catch (Exception)

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public bool DeleteMember(string teamName, AllUserAndTeam member)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

cvDatabase.TeamMembers.DeleteObject(cvDatabase.TeamMembers.First(x => x.UserID.Equals(member.Name) && x.Name.Equals(teamName)));

cvDatabase.SaveChanges();

return true;

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public counterSetting CounterSettings()

{

try

{

string counterConfigXmlpath = Path.Combine(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "Cafeteria\_Vernier"), "countConfig.xml");

if (!File.Exists(counterConfigXmlpath))

{

var counterConfig = XDocument.Load(counterConfigXmlpath);

counterSetting countSetting = (from countConfig in counterConfig.Descendants("MunitiesCount") select new counterSetting { seconds = (int)countConfig.Attribute("second") }).SingleOrDefault();

return countSetting;

}

else

{

return new counterSetting { seconds = 60 };

}

}

catch (Exception error)

{

throw new FaultException(error.Message);

}

}

public byte[] GetUserImage(string username)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

return cvDatabase.CustomerInformations.First(x => x.UserID.Equals(username)).Logo;

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0,2]);

}

}

public List<AllUserAndTeam> GetTeamName(string adminName)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

return (from teamTable in cvDatabase.Teams

where teamTable.AdminName.Equals(adminName)

select new AllUserAndTeam

{

Name=teamTable.Name.Trim()

}).ToList();

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public bool UpdateTeam(string teamName, List<AllUserAndTeam> teamMembers)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

foreach (var existingMember in cvDatabase.TeamMembers.Where(x=>x.Name.Equals(teamName)))

{

cvDatabase.TeamMembers.DeleteObject(existingMember);

}

foreach (AllUserAndTeam teamMember in teamMembers)

{

cvDatabase.AddToTeamMembers(new TeamMember { Name = teamName, UserID = teamMember.Name });

}

cvDatabase.SaveChanges();

return true;

}

}

catch (Exception error)

{

throw new FaultException(error.Message);

}

}

public List<UserRechareHistory> TeamRechargeHistoryAll(string TeamName)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase= new Cafeteria\_Vernier\_dbEntities())

{

return (from rechargeHisTable in cvDatabase.TeamRechargeHistories where rechargeHisTable.Name.Equals(TeamName) select new UserRechareHistory

{

Amount=rechargeHisTable.bill,

EmployID=rechargeHisTable.EmployeeID,

EntryDate=rechargeHisTable.DateWithTime,

Munities=rechargeHisTable.Minutes

}).ToList();

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public List<UserRechareHistory> TeamRechargeHistoryDate(string TeamName, DateTime date)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

return (from rechargeHisTable in cvDatabase.TeamRechargeHistories

where rechargeHisTable.Name.Equals(TeamName) && rechargeHisTable.DateWithTime ==date

select new UserRechareHistory

{

Amount = rechargeHisTable.bill,

EmployID = rechargeHisTable.EmployeeID,

EntryDate = rechargeHisTable.DateWithTime,

Munities = rechargeHisTable.Minutes

}).ToList();

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public List<UserRechareHistory> TeamRechargeHistoryTwoDate(string TeamName, DateTime firstDate, DateTime secondDate)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

return (from rechargeHisTable in cvDatabase.TeamRechargeHistories

where rechargeHisTable.Name.Equals(TeamName) && rechargeHisTable.DateWithTime<=firstDate && rechargeHisTable.DateWithTime>=secondDate

select new UserRechareHistory

{

Amount = rechargeHisTable.bill,

EmployID = rechargeHisTable.EmployeeID,

EntryDate = rechargeHisTable.DateWithTime,

Munities = rechargeHisTable.Minutes

}).ToList();

}

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public bool UserLogout(string username, Int64 loginHistoryID)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

var userinfo = cvDatabase.CustomerAccounts.FirstOrDefault(x => x.UserID.Equals(username));

if (userinfo!=null)

{

userinfo.Status = false;

userinfo.Counternumber = 0;

cvDatabase.UserLoginHistories.First(x => x.AutoInc == loginHistoryID).EndTime = DateTime.Now;

cvDatabase.SaveChanges();

}

else

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 3]);

}

}

return true;

}

catch

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public bool TeamLogout(string username, string teamName, Int64 loginHistoryID)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

var userInfo = cvDatabase.CustomerAccounts.FirstOrDefault(x => x.UserID.Equals(username));

var teamInfo = cvDatabase.TeamAccounts.FirstOrDefault(x => x.Name.Equals(teamName));

if (userInfo !=null && teamInfo!=null)

{

userInfo.Status = false;

userInfo.Counternumber = 0;

teamInfo.Status = false;

cvDatabase.UserLoginHistories.First(x => x.AutoInc == loginHistoryID).EndTime = DateTime.Now;

cvDatabase.SaveChanges();

return true;

}

else

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 3]);

}

}

}

catch (Exception)

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public List<AllUserAndTeam> GetTeamMember(string teamName)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

return (from teamMember in cvDatabase.TeamMembers where teamMember.Name.Equals(teamName) select new AllUserAndTeam

{

Name=teamMember.UserID.Trim()

}).ToList();

}

}

catch (Exception)

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 4]);

}

}

public int GetUserBalance(string username)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

var userInfo = cvDatabase.CustomerAccounts.FirstOrDefault(x => x.UserID.Equals(username));

if (userInfo!=null)

{

return userInfo.Minutes;

}

else

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

}

catch (Exception)

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public int GetTeamBalance(string teamName)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

var teamInfo = cvDatabase.TeamAccounts.FirstOrDefault(x => x.Name.Equals(teamName));

if (teamInfo!=null)

{

return teamInfo.Minutes;

}

else

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

}

catch (Exception)

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public bool PasswordChange(string username, string newPassword)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

var customerInfo = cvDatabase.CustomerAccounts.FirstOrDefault(x => x.UserID.Equals(username));

if (customerInfo!=null)

{

customerInfo.Password = newPassword;

cvDatabase.SaveChanges();

return true;

}

else

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

}

catch (Exception)

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public byte[] GetTeamLogo(string teamName)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

return cvDatabase.Teams.First(x => x.Name.Equals(teamName)).Logo;

}

}

catch (Exception)

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

public void UpdateTeamLogo(string teamName, byte[] teamLogo)

{

try

{

using (Cafeteria\_Vernier\_dbEntities cvDatabase = new Cafeteria\_Vernier\_dbEntities())

{

cvDatabase.Teams.First(x => x.Name.Equals(teamName)).Logo = teamLogo;

}

}

catch (Exception)

{

throw new FaultException(ServiceVariables.ERROR\_MESSAGES[0, 2]);

}

}

}

}

Client Code

**Class: UserInfoViewer**

namespace Procesta.CVClient.Class.CvCPropertes

{

public class UserInfoViewer :INotifyPropertyChanged

{

private Int64 \_Minutes;

private byte[] \_Photo;

private string \_Username;

private string \_Password;

private string \_TeamName = string.Empty;

private Int64 \_LoginHistoryID;

public Int64 Minutes

{

get { return this.\_Minutes; }

set { this.\_Minutes = value; this.OnPropertyChange("Minutes"); }

}

public byte[] Photo

{

get { return this.\_Photo; }

set { this.\_Photo = value; this.OnPropertyChange("Photo"); }

}

public string Username

{

get { return this.\_Username; }

set { this.\_Username = value; this.OnPropertyChange("Username"); }

}

public string Password

{

get { return this.\_Password; }

set { this.\_Password = value; this.OnPropertyChange("Password"); }

}

public string TeamName

{

get { return this.\_TeamName; }

set { this.\_TeamName = value; this.OnPropertyChange("TeamName"); }

}

public Int64 LoginHistoryID

{

get { return this.\_LoginHistoryID; }

set { this.\_LoginHistoryID = value; this.OnPropertyChange("LoginHistoryID"); }

}

public event PropertyChangedEventHandler PropertyChanged;

private void OnPropertyChange(string propertyName)

{

if (PropertyChanged!=null)

{

this.PropertyChanged(this,new PropertyChangedEventArgs(propertyName));

}

}

}

}

**Class : ServerConnection**

namespace Procesta.CVClient.Class.Methords

{

public class ServerConnection

{

public static ServerSideServicesClient serviceFromServer = null;

public static ClientNotificationClient clientNotifaction = null;

public static ServerNotificationClient serverNotifaction = null;

/// <summary>

/// Connect To Server Services

/// </summary>

/// <returns></returns>

public static bool ConnectToService()

{

try

{

serviceFromServer = new ServerSideServicesClient("NetTcpBinding\_IServerSideServices");

return true;

}

catch

{

return false;

}

}

/// <summary>

/// Connect and Send Information to Server

/// </summary>

public static void NotifactionFromServer()

{

try

{

clientNotifaction = new ClientNotificationClient("NetTcpBinding\_IClientNotification");

string counterNumber=Properties.Settings.Default.CounterNumber.ToString();

new Task(

new Action(() =>

{

while (true)

{

try

{

ObservableCollection<CommandData> commands = new ObservableCollection<CommandData>(clientNotifaction.GetCommands(counterNumber));

if (commands.Count > 0)

{

foreach (CommandData comm in commands)

{

clientNotifaction.RemoveCommand(counterNumber);

switch (comm.Command)

{

case Commands.Shutdown:

Process.Start("shutdown", "/s /t 0");

break;

case Commands.Restart:

Process.Start("shutdown", "/r /t 0");

break;

case Commands.AccountLogout:

Counter.counterWindow.Dispatcher.BeginInvoke(new Action(Counter.counterWindow.Close), DispatcherPriority.Normal);

break;

default:

break;

}

}

}

System.Threading.Thread.Sleep(6000);

}

catch

{

break;

}

}

})).Start();

}

catch

{

System.Threading.Thread.Sleep(10000);

NotifactionFromServer();

}

}

public static void NotifactionToServer()

{

try

{

serverNotifaction = new ServerNotificationClient("NetTcpBinding\_IServerNotification");

MiraculousMethods.conterInformation.CounterName = Properties.Settings.Default.CounterName;

MiraculousMethods.conterInformation.CounterNumber = Properties.Settings.Default.CounterNumber.ToString();

MiraculousMethods.conterInformation.CustomerID = null;

MiraculousMethods.conterInformation.Status = CounterStatus.Free;

new Task(new Action(() =>

{

while (true)

{

try

{

MiraculousMethods.conterInformation.sendingTime = DateTime.Now;

serverNotifaction.SetCounterInformation(MiraculousMethods.conterInformation);

System.Threading.Thread.Sleep(5000);

}

catch

{

break;

}

}

})).Start();

}

catch

{

System.Threading.Thread.Sleep(10000);

NotifactionToServer();

}

}

/// <summary>

/// Check A Server Alive or Not

/// </summary>

/// <param name="serviceChannel"></param>

/// <returns></returns>

public static bool ServerAliveIs(ICommunicationObject serviceChannel)

{

if (serviceChannel.State.Equals(CommunicationState.Closed))

{

return false;

}

else if(serviceChannel.State.Equals(CommunicationState.Closing))

{

return false;

}

else if (serviceChannel.State.Equals(CommunicationState.Faulted))

{

return false;

}

else

{

return true;

}

}

}

}

Class: IPAddressValidator

namespace Procesta.CVClient.Class.ValidationRuls

{

public class IPAddressValidator :ValidationRule

{

public override ValidationResult Validate(object value, System.Globalization.CultureInfo cultureInfo)

{

if (value!=null)

{

Regex ipAddressRegex = new Regex(@"\b(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b");

if (ipAddressRegex.Match(value.ToString()).Success)

{

return new ValidationResult(true, string.Empty);

}

else

{

return new ValidationResult(false, "Please enter a valid IP Address");

}

}

else

{

return new ValidationResult(true,string.Empty);

}

}

}

}

**Class: TeamNameValidator**

namespace Procesta.CVClient.Class.ValidationRuls

{

public class TeamNameValidator: ValidationRule

{

public override ValidationResult Validate(object value, System.Globalization.CultureInfo cultureInfo)

{

string teamName = (string)value;

if (teamName==null || string.IsNullOrEmpty(teamName) || string.IsNullOrWhiteSpace(teamName))

{

return new ValidationResult(false, "Please enter a team name");

}

if (ServerConnection.serviceFromServer.TeamNameChecker(teamName))

{

return new ValidationResult(true, "");

}

else

{

return new ValidationResult(false, "Team Name already exist");

}

}

}

}

**Class: counterForgroudConverter**

namespace Procesta.CVClient.Class.ValueConverter

{

public class counterForgroudConverter: IValueConverter

{

public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

LinearGradientBrush counterForgroundBrush = new LinearGradientBrush();

counterForgroundBrush.StartPoint = new System.Windows.Point(0.5, 0);

counterForgroundBrush.EndPoint = new System.Windows.Point(0.5, 1);

if (value != null)

{

int munities=(int)value;

if (munities>=20)

{

counterForgroundBrush.GradientStops.Add(new GradientStop((Color)ColorConverter.ConvertFromString("#FF299503"), 0));

counterForgroundBrush.GradientStops.Add(new GradientStop((Color)ColorConverter.ConvertFromString("#FF94EB75"), 1));

counterForgroundBrush.GradientStops.Add(new GradientStop((Color)ColorConverter.ConvertFromString("#FF05AF17"), 0.5));

return counterForgroundBrush;

}

else if(munities>=10)

{

counterForgroundBrush.GradientStops.Add(new GradientStop((Color)ColorConverter.ConvertFromString("#FFD5D600"), 0));

counterForgroundBrush.GradientStops.Add(new GradientStop((Color)ColorConverter.ConvertFromString("#FFEBDC75"), 1));

counterForgroundBrush.GradientStops.Add(new GradientStop((Color)ColorConverter.ConvertFromString("#FFB5B600"), 0.5));

return counterForgroundBrush;

}

else

{

counterForgroundBrush.GradientStops.Add(new GradientStop((Color)ColorConverter.ConvertFromString("#FFD01010"), 0));

counterForgroundBrush.GradientStops.Add(new GradientStop((Color)ColorConverter.ConvertFromString("#FFE44343"), 1));

counterForgroundBrush.GradientStops.Add(new GradientStop((Color)ColorConverter.ConvertFromString("#FFB60000"), 0.5));

return counterForgroundBrush;

}

}

else

{

counterForgroundBrush.GradientStops.Add(new GradientStop((Color)ColorConverter.ConvertFromString("#FF299503"), 0));

counterForgroundBrush.GradientStops.Add(new GradientStop((Color)ColorConverter.ConvertFromString("#FF94EB75"), 1));

counterForgroundBrush.GradientStops.Add(new GradientStop((Color)ColorConverter.ConvertFromString("#FF05AF17"), 0.5));

return counterForgroundBrush;

}

}

public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

Class: Counter

namespace Procesta.CVClient

{

/// <summary>

/// Interaction logic for Counter.xaml

/// </summary>

public partial class Counter : Window, INotifyPropertyChanged

{

#region Property

private UserInfoViewer \_customerDetail;

public UserInfoViewer customerDetail

{

get

{

return this.\_customerDetail;

}

set

{

this.\_customerDetail = value;

if (this.PropertyChanged != null)

{

this.PropertyChanged(this, new PropertyChangedEventArgs("customerDetail"));

}

}

}

public static Window counterWindow = null;

#endregion

#region Private Variables

DispatcherTimer timeCounter = new DispatcherTimer();

short counterNumber;

#endregion

#region Constructor

/// <summary>

/// Default Constructor

/// </summary>

public Counter()

{

customerDetail = new UserInfoViewer();

this.InitializeComponent();

this.Top = System.Windows.Forms.Screen.PrimaryScreen.Bounds.Height - (this.Height + 50);

this.Left = System.Windows.Forms.Screen.PrimaryScreen.Bounds.Width - (this.Width + 10);

timeCounter.Interval = new TimeSpan(0, 0, 59);

counterNumber = Properties.Settings.Default.CounterNumber;

timeCounter.Start();

this.Closing += new System.ComponentModel.CancelEventHandler(Counter\_Closing);

counterWindow = this;

}

void Counter\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

this.timeCounter.Stop();

new MainWindow().Show();

this.LogoutOperation();

MiraculousMethods.conterInformation.CustomerID = null;

MiraculousMethods.conterInformation.Status = CounterStatus.Free;

customerDetail = null;

}

/// <summary>

/// With User Information

/// </summary>

/// <param name="userinfo"></param>

public static void CounterWindow(UserInfoViewer loggedInUserInfo)

{

Counter counterWindow = new Counter();

counterWindow.customerDetail = loggedInUserInfo;

MiraculousMethods.conterInformation.CustomerID = loggedInUserInfo.Username;

MiraculousMethods.conterInformation.Status = CounterStatus.Busy;

if (String.IsNullOrEmpty(loggedInUserInfo.TeamName) || String.IsNullOrWhiteSpace(loggedInUserInfo.TeamName))

{

counterWindow.userLoginTimer();

}

else

{

counterWindow.teamLoginTimer();

}

counterWindow.Show();

}

#endregion

#region Timer Handelar

/// <summary>

/// User munities decrement

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void UserTimeDecriment(object sender, EventArgs e)

{

ConnectionChecker();

if (ServerConnection.serviceFromServer.UserMunitieCounter(customerDetail.Username, counterNumber))

{

customerDetail.Minutes = ServerConnection.serviceFromServer.GetUserBalance(customerDetail.Username);

}

else

{

customerDetail.Minutes = ServerConnection.serviceFromServer.GetUserBalance(customerDetail.Username);

this.TimeChecker();

}

}

/// <summary>

/// Team munities decrement

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void TeamTimeDecriment(object sender, EventArgs e)

{

ConnectionChecker();

if (ServerConnection.serviceFromServer.TeamMunitieCounter(customerDetail.TeamName, customerDetail.Username, counterNumber))

{

customerDetail.Minutes = ServerConnection.serviceFromServer.GetTeamBalance(customerDetail.TeamName);

}

else

{

customerDetail.Minutes = ServerConnection.serviceFromServer.GetTeamBalance(customerDetail.TeamName);

this.TimeChecker();

}

}

#endregion

#region Buttons

public ICommand LogoutCommand

{

get { return new ReplayCommand(new Action<object>(this.logout)); }

}

public ICommand SettingCommand

{

get { return new ReplayCommand(new Action<object>(this.setting)); }

}

private void logout(object obj)

{

this.btnLogout.IsEnabled = false;

try

{

if (DXMessageBox.Show(CVsVariables.ERROR\_MESSAGES[0, 3], CVsVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.YesNo, MessageBoxImage.Question).Equals(MessageBoxResult.Yes))

{

//Mouse.OverrideCursor = Cursors.Wait;

//this.timeCounter.Stop();

//new MainWindow().Show();

//this.LogoutOperation();

//customerDetail = null;

this.Close();

}

}

catch (Exception ex)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(ex.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.btnLogout.IsEnabled = true;

}

}

private void setting(object obj)

{

this.btnSetting.IsEnabled = false;

try

{

Settings.counter = this;

new Settings().ShowDialog();

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.btnSetting.IsEnabled = true;

}

}

#endregion

#region Private Mentod

/// <summary>

/// Check Time is >1 or not

/// </summary>

private void TimeChecker()

{

if (customerDetail.Minutes <= 1)

{

this.timeCounter.Stop();

new TimeOut().Show();

this.LogoutOperation();

customerDetail = null;

this.Close();

}

}

/// <summary>

/// Timer Handle change for user login

/// </summary>

private void userLoginTimer()

{

timeCounter.Tick -= new EventHandler(TeamTimeDecriment);

timeCounter.Tick += new EventHandler(UserTimeDecriment);

}

/// <summary>

/// Timer handle change for team login

/// </summary>

private void teamLoginTimer()

{

timeCounter.Tick += new EventHandler(TeamTimeDecriment);

timeCounter.Tick -= new EventHandler(UserTimeDecriment);

}

/// <summary>

/// Logout Operation

/// </summary>

private void LogoutOperation()

{

try

{

this.ConnectionChecker();

if (customerDetail.TeamName != string.Empty && customerDetail.TeamName != null)

{

ServerConnection.serviceFromServer.TeamLogout(username: customerDetail.Username, teamName: customerDetail.TeamName, loginHistoryID: customerDetail.LoginHistoryID);

}

else

{

ServerConnection.serviceFromServer.UserLogout(customerDetail.Username,customerDetail.LoginHistoryID);

}

}

catch (Exception error)

{

DXMessageBox.Show(error.Message, CVsVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

}

}

/// <summary>

/// Check Server Connection Is Alive or not

/// </summary>

private void ConnectionChecker()

{

if (ServerConnection.serviceFromServer == null || !ServerConnection.ServerAliveIs((ICommunicationObject)ServerConnection.serviceFromServer))

{

ServerConnection.ConnectToService();

}

}

#endregion

public static void counterLogout()

{

if (counterWindow!=null)

{

}

}

public event PropertyChangedEventHandler PropertyChanged;

}

}

Class: InstallWindow

namespace Procesta.CVClient

{

/// <summary>

/// Interaction logic for InstallWindow.xaml

/// </summary>

public partial class InstallWindow : Window

{

public InstallWindow()

{

InitializeComponent();

}

public ICommand UpdateButtonCommand

{

get

{

return new ReplayCommand(new Action<object>(this.UpdateButton\_Click));

}

}

private void UpdateButton\_Click(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

Properties.Settings.Default.IsConfigered = true;

System.Windows.Forms.Application.Restart();

System.Windows.Application.Current.Shutdown();

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

}

}

/// <summary>

/// Window Load Event

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void IpAddressWindowsLoaded(object sender, System.Windows.RoutedEventArgs e)

{

IPAddress[] localPs = Dns.GetHostAddresses(Dns.GetHostName());

foreach (IPAddress WorkingIp in localPs)

{

this.IpAddressComBoBox.Items.Add(WorkingIp);

}

}

}

}

Class: MainWindow

namespace Procesta.CVClient

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window, INotifyPropertyChanged

{

private bool \_IsBusy;

public bool IsBusy

{

get { return this.\_IsBusy; }

set

{

this.\_IsBusy = value;

onPropertyChnage("IsBusy");

}

}

public MainWindow()

{

InitializeComponent();

this.LogtextboxUserName.Focus();

this.PanelLogin.DataContext = new UserInfoViewer();

}

private void testCloseClick(object sender, System.Windows.RoutedEventArgs e)

{

Application.Current.Shutdown();

}

#region Panel Login

public ICommand CustomerLoginCommand

{

get { return new ReplayCommand(new Action<object>(this.customerLogin)); }

}

private void customerLogin(object obj)

{

this.IsBusy = true;

this.CustomerLogin.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

UserInfoViewer loggedInCutomer = obj as UserInfoViewer;

if (String.IsNullOrEmpty(loggedInCutomer.Username) || String.IsNullOrWhiteSpace(loggedInCutomer.Username) || String.IsNullOrEmpty(loggedInCutomer.Password) || string.IsNullOrWhiteSpace(loggedInCutomer.Password))

{

DXMessageBox.Show("Please enter Username and Password", CVsVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (ServerConnection.serviceFromServer == null || !ServerConnection.ServerAliveIs((ICommunicationObject)ServerConnection.serviceFromServer))

{

ServerConnection.ConnectToService();

}

if (String.IsNullOrEmpty(loggedInCutomer.TeamName) || String.IsNullOrWhiteSpace(loggedInCutomer.TeamName))

{

List<Int64> userInformation = ServerConnection.serviceFromServer.UserLogin(loggedInCutomer.Username, loggedInCutomer.Password, Properties.Settings.Default.CounterNumber);

loggedInCutomer.Minutes = userInformation.First();

loggedInCutomer.LoginHistoryID = userInformation.Last();

}

else

{

List<Int64> teamInformation = ServerConnection.serviceFromServer.TeamLogin(loggedInCutomer.Username, loggedInCutomer.Password, Properties.Settings.Default.CounterNumber, loggedInCutomer.TeamName);

loggedInCutomer.Minutes = teamInformation.First();

loggedInCutomer.LoginHistoryID = teamInformation.Last();

}

loggedInCutomer.Photo = ServerConnection.serviceFromServer.GetUserImage(loggedInCutomer.Username);

Counter.CounterWindow(loggedInCutomer);

this.Close();

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

this.CustomerLogin.IsEnabled = true;

Mouse.OverrideCursor = null;

this.IsBusy = false;

}

}

#endregion

#region Panel shudown

/// <summary>

/// Restart Button Click

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void LogImageButtonRestartClcik(object sender, System.Windows.RoutedEventArgs e)

{

try

{

Process.Start("shutdown","/r /t 0");

}

catch (Exception error)

{

DXMessageBox.Show(error.Message, CVsVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

}

}

/// <summary>

/// Shutdown Button Click

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void LogImageButtonShutdownClick(object sender, System.Windows.RoutedEventArgs e)

{

try

{

Process.Start("shutdown","/s /t 0");

}

catch (Exception error)

{

DXMessageBox.Show(error.Message, CVsVariables.ERROR\_MESSAGES[0, 0], MessageBoxButton.OK, MessageBoxImage.Error);

}

}

#endregion

private void onPropertyChnage(string propertyName)

{

if (this.PropertyChanged!=null)

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

public event PropertyChangedEventHandler PropertyChanged;

}

}

Class: Setting

namespace Procesta.CVClient

{

/// <summary>

/// Interaction logic for Settings.xaml

/// </summary>

public partial class Settings : Window, INotifyPropertyChanged

{

#region Private Variables

public ObservableCollection<AllUserAndTeam> \_newMembers;

private string \_Option;

private bool \_IsBusy;

public string Option

{

get

{

return this.\_Option;

}

set

{

this.\_Option = value;

OnPropertyChanged("Option");

}

}

public static Counter counter;

public ObservableCollection<AllUserAndTeam> newMembers

{

get

{

return this.\_newMembers;

}

set

{

this.\_newMembers = value;

OnPropertyChanged("newMembers");

}

}

public bool IsBusy

{

get

{

return this.\_IsBusy;

}

set

{

this.\_IsBusy = value;

OnPropertyChanged("IsBusy");

}

}

private List<AllUserAndTeam> AddRemovemembers = new List<AllUserAndTeam>();

#endregion

public Settings()

{

newMembers = new ObservableCollection<AllUserAndTeam>();

this.InitializeComponent();

Mouse.OverrideCursor = null;

this.DataContext = this;

}

#region Memu Bar

public ICommand MenuBarCommand

{

get { return new ReplayCommand(new Action<object>(this.menuBarButton\_Click)); }

}

private void menuBarButton\_Click(object obj)

{

try

{

this.HideAllPanel();

this.Option = "ByDate";

switch (obj as string)

{

case "ProfileEdit":

new Task(new Action(() =>

{

this.ServiceConnectionChecker();

this.Dispatcher.BeginInvoke(new Action(() =>

{

this.PanelCustInfoEdit.DataContext = ServerConnection.serviceFromServer.AccountDetails(counter.customerDetail.Username);

}));

})).Start();

this.PanelCustInfoEdit.Visibility = Visibility.Visible;

break;

case "ChangePassword":

this.PasswordChangeClear();

this.settingPanelChangePassword.Visibility = Visibility.Visible;

break;

case "RechargeHistory":

this.RechHisGridView.ItemsSource = null;

this.panelRechHis.Visibility = Visibility.Visible;

break;

case "UserLoginHistory":

this.logHisGridView.ItemsSource = null;

this.PanelLoginHistory.Visibility = Visibility.Visible;

break;

case "AddTeam":

TeamInfo newTeamInfo = new TeamInfo();

this.PanelNewTeam.DataContext = newTeamInfo;

new Task(new Action(() =>

{

this.ServiceConnectionChecker();

this.Dispatcher.BeginInvoke(new Action(() =>

{

this.teamUserList.ItemsSource = ServerConnection.serviceFromServer.GetAllUsers();

}));

})).Start();

this.teamMemberList.ItemsSource = new ObservableCollection<AllUserAndTeam>();

this.PanelNewTeam.Visibility = Visibility.Visible;

break;

case "EditTeam":

new Task(new Action(() =>

{

this.ServiceConnectionChecker();

this.Dispatcher.BeginInvoke(new Action(() =>

{

this.editTeamComboBoxTeamName.ItemsSource = ServerConnection.serviceFromServer.GetTeamName(counter.customerDetail.Username);

this.teamEditUserList.ItemsSource = ServerConnection.serviceFromServer.GetAllUsers();

}));

})).Start();

this.PanelEditTeam.Visibility = Visibility.Visible;

break;

case "TeamRechargeHistory":

this.RechHisGridView.ItemsSource = null;

this.TeamRechHisTeamName.ItemsSource = ServerConnection.serviceFromServer.GetTeamName(counter.customerDetail.Username);

this.PanelTeamRechHis.Visibility = Visibility.Visible;

break;

default:

break;

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

}

#endregion

#region Panel Profile Edit

public ICommand ProfileImageBrowseCommand

{

get { return new ReplayCommand(new Action<object>(this.profileImagebrowseClick)); }

}

public ICommand ProfileUpdateCommand

{

get { return new ReplayCommand(new Action<object>(this.profileUpdateClick)); }

}

private void profileImagebrowseClick(object obj)

{

this.profileImageBrowse.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

Userinformation userInfo = obj as Userinformation;

userInfo.UserImage = imageOpenDialogBox();

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

this.profileImageBrowse.IsEnabled = true;

Mouse.OverrideCursor = null;

}

}

private void profileUpdateClick(object obj)

{

this.profileUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

Userinformation userInfo = obj as Userinformation;

this.ServiceConnectionChecker();

if (ServerConnection.serviceFromServer.AccountUpdate(counter.customerDetail.Username, userInfo))

{

counter.customerDetail.Photo = userInfo.UserImage;

Mouse.OverrideCursor = null;

DXMessageBox.Show(CVsVariables.ERROR\_MESSAGES[0, 6], CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CVsVariables.ERROR\_MESSAGES[0, 7], CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

this.profileUpdate.IsEnabled = true;

Mouse.OverrideCursor = null;

}

}

#endregion

#region Panel User Login History

public ICommand LoginHistorySearchCommand

{

get { return new ReplayCommand(new Action<object>(this.loginHistorySearch\_Click)); }

}

private void loginHistorySearch\_Click(object obj)

{

this.logHisButtonSearch.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

this.IsBusy = true;

try

{

if (obj is ArrayList)

{

ArrayList dataList = obj as ArrayList;

this.ServiceConnectionChecker();

switch (this.Option)

{

case "ByDate":

this.logHisGridView.ItemsSource = ServerConnection.serviceFromServer.UserLoginHistoryDate(counter.customerDetail.Username, (DateTime)dataList[0]);

break;

case "BetweenTwoDate":

this.logHisGridView.ItemsSource = ServerConnection.serviceFromServer.UserLoginHistoryTwoDate(counter.customerDetail.Username, (DateTime)dataList[1], (DateTime)dataList[2]);

break;

case "All":

this.logHisGridView.ItemsSource = ServerConnection.serviceFromServer.UserLoginHistoryAll(counter.customerDetail.Username);

break;

default:

break;

}

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

this.logHisButtonSearch.IsEnabled = true;

Mouse.OverrideCursor = null;

this.IsBusy = false;

}

}

#endregion

#region Panel User Recharge History

public ICommand UserRechHisCommnad

{

get { return new ReplayCommand(new Action<object>(this.userRechHis\_Click)); }

}

private void userRechHis\_Click(object obj)

{

this.RechHisButtonSearch.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

this.IsBusy = true;

try

{

if (obj is ArrayList)

{

ArrayList dataList = obj as ArrayList;

this.ServiceConnectionChecker();

switch (this.Option)

{

case "ByDate":

this.RechHisGridView.ItemsSource = ServerConnection.serviceFromServer.UserRechargeHistoryDate(counter.customerDetail.Username, (DateTime)dataList[0]);

break;

case "BetweenTwoDate":

this.RechHisGridView.ItemsSource = ServerConnection.serviceFromServer.UserRechargeHistoryTwoDate(counter.customerDetail.Username, (DateTime)dataList[1], (DateTime)dataList[2]);

break;

case "All":

this.RechHisGridView.ItemsSource = ServerConnection.serviceFromServer.UserRechargeHistoryAll(counter.customerDetail.Username);

break;

default:

break;

}

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

this.RechHisButtonSearch.IsEnabled = true;

Mouse.OverrideCursor = null;

this.IsBusy = false;

}

}

#endregion

#region Panel New Team

public ICommand TeamImageBrowseCommand

{

get { return new ReplayCommand(new Action<object>(this.teamImageBrowse\_Click)); }

}

public ICommand TeamMemberAddCommand

{

get { return new ReplayCommand(new Action<object>(this.memberAdd\_Click)); }

}

public ICommand TeamMemberRemoveCommand

{

get { return new ReplayCommand(new Action<object>(this.memberRemove\_Click)); }

}

public ICommand TeamUpdateCommand

{

get { return new ReplayCommand(new Action<object>(this.teamUpdate\_Click)); }

}

private void teamImageBrowse\_Click(object obj)

{

this.teamImagebrowse.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is TeamInfo)

{

TeamInfo newTeamInfo = obj as TeamInfo;

newTeamInfo.TeamImage = this.imageOpenDialogBox();

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

this.teamImagebrowse.IsEnabled = true;

Mouse.OverrideCursor = null;

}

}

private void memberAdd\_Click(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is AllUserAndTeam)

{

AllUserAndTeam newTeamMember = obj as AllUserAndTeam;

var teamMembers = this.teamMemberList.ItemsSource as ObservableCollection<AllUserAndTeam>;

if (teamMembers.FirstOrDefault(x => x.Name.Equals(newTeamMember.Name)) == null)

{

teamMembers.Add(newTeamMember);

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show("This user is already a member.", CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Stop);

}

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

}

}

private void memberRemove\_Click(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is AllUserAndTeam)

{

(this.teamMemberList.ItemsSource as ObservableCollection<AllUserAndTeam>).Remove(obj as AllUserAndTeam);

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

}

}

/// <summary>

///

/// </summary>

/// <param name="obj">

/// obj[0] = TeamInfo

/// obj[1] = AllUserAndTeam

/// </param>

private void teamUpdate\_Click(object obj)

{

this.teamUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is ArrayList)

{

this.ServiceConnectionChecker();

ArrayList dataList = obj as ArrayList;

TeamInfo newTeamInfo= dataList[0] as TeamInfo;

newTeamInfo.TeamAdmin=counter.customerDetail.Username;

if (ServerConnection.serviceFromServer.AddNewTeam(newTeamInfo, new List<AllUserAndTeam>(dataList[1] as ObservableCollection<AllUserAndTeam>)))

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(CVsVariables.ERROR\_MESSAGES[0, 10], CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

this.teamUpdate.IsEnabled = true;

Mouse.OverrideCursor = null;

}

}

#endregion

#region Panel Edit Team

public ICommand GetTeamInfoCommand

{

get { return new ReplayCommand(new Action<object>(this.getTeamInfo\_SelectionChange)); }

}

public ICommand TeamEditAddCommand

{

get { return new ReplayCommand(new Action<object>(this.teamEditMemberAdd\_Click)); }

}

public ICommand TeamEditRemoveCommand

{

get { return new ReplayCommand(new Action<object>(this.teamEditMemberDelete\_Click)); }

}

public ICommand TeamEditImageBrowseCommand

{

get { return new ReplayCommand(new Action<object>(this.teamEditImageBrowse\_Click)); }

}

public ICommand TeamEditUpdateCommand

{

get { return new ReplayCommand(new Action<object>(this.teamEditUpdate\_Click)); }

}

private void getTeamInfo\_SelectionChange(object obj)

{

this.IsBusy = true;

Mouse.OverrideCursor = Cursors.Wait;

try

{

string teamName = obj as String;

this.ServiceConnectionChecker();

teamEditMemberList.ItemsSource =new ObservableCollection<AllUserAndTeam>(ServerConnection.serviceFromServer.GetTeamMember(teamName));

TeamInfo teamInformation = new TeamInfo();

teamInformation.TeamImage = ServerConnection.serviceFromServer.GetTeamLogo(teamName);

this.editTeamImageBorder.DataContext = teamInformation;

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

this.IsBusy = false;

Mouse.OverrideCursor = null;

}

}

private void teamEditMemberAdd\_Click(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is AllUserAndTeam)

{

AllUserAndTeam newTeamMember = obj as AllUserAndTeam;

newTeamMember.AddOrDelete = true;

var teamMembers = this.teamEditMemberList.ItemsSource as ObservableCollection<AllUserAndTeam>;

if (teamMembers.FirstOrDefault(x => x.Name.Equals(newTeamMember.Name)) == null)

{

teamMembers.Add(newTeamMember);

}

else

{

Mouse.OverrideCursor = null;

DXMessageBox.Show("This user is already a member.", CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Stop);

}

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

}

}

private void teamEditMemberDelete\_Click(object obj)

{

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is AllUserAndTeam)

{

var teamMember = obj as AllUserAndTeam;

teamMember.AddOrDelete = false;

(this.teamEditMemberList.ItemsSource as ObservableCollection<AllUserAndTeam>).Remove(teamMember);

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

}

}

private void teamEditImageBrowse\_Click(object obj)

{

this.editTeamBrowse.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is TeamInfo)

{

TeamInfo newTeamInfo = obj as TeamInfo;

newTeamInfo.TeamImage = this.imageOpenDialogBox();

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

this.editTeamBrowse.IsEnabled = true;

Mouse.OverrideCursor = null;

}

}

private void teamEditUpdate\_Click(object obj)

{

this.teamEditUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

if (obj is ArrayList)

{

ArrayList dataList = obj as ArrayList;

this.ServiceConnectionChecker();

if (ServerConnection.serviceFromServer.UpdateTeam(dataList[0] as String, new List<AllUserAndTeam>(dataList[1] as ObservableCollection<AllUserAndTeam>)))

{

this.AddRemovemembers.Clear();

Mouse.OverrideCursor = null;

DXMessageBox.Show(CVsVariables.ERROR\_MESSAGES[0, 6], CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

this.teamEditUpdate.IsEnabled = true;

Mouse.OverrideCursor = null;

}

}

#endregion

#region Panel Change Password

public ICommand ChangePasswordCommand

{

get { return new ReplayCommand(new Action<object>(this.changePassword\_Click)); }

}

private void changePassword\_Click(object obj)

{

this.passwordChangeUpdate.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

try

{

this.ServiceConnectionChecker();

if (ServerConnection.serviceFromServer.PasswordChange(counter.customerDetail.Username, obj as String))

{

MiraculasProperty.password = obj as String;

DXMessageBox.Show(CVsVariables.ERROR\_MESSAGES[0, 6], CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

Mouse.OverrideCursor = null;

this.passwordChangeUpdate.IsEnabled = true;

}

}

#endregion

#region panel Team Rechage History

public ICommand TeamRechHisCommnad

{

get { return new ReplayCommand(new Action<object>(this.TeamRechHis\_Click)); }

}

private void TeamRechHis\_Click(object obj)

{

this.TeamRechHistorySearch.IsEnabled = false;

Mouse.OverrideCursor = Cursors.Wait;

this.IsBusy = true;

try

{

if (obj is ArrayList)

{

ArrayList dataList = obj as ArrayList;

this.ServiceConnectionChecker();

switch (this.Option)

{

case "ByDate":

this.TeamRechHisGridView.ItemsSource = ServerConnection.serviceFromServer.TeamRechargeHistoryDate(dataList[3] as String, (DateTime)dataList[0]);

break;

case "BetweenTwoDate":

this.TeamRechHisGridView.ItemsSource = ServerConnection.serviceFromServer.TeamRechargeHistoryTwoDate(dataList[3] as String, (DateTime)dataList[1], (DateTime)dataList[2]);

break;

case "All":

this.TeamRechHisGridView.ItemsSource = ServerConnection.serviceFromServer.TeamRechargeHistoryAll(dataList[3] as String);

break;

default:

break;

}

}

}

catch (Exception error)

{

Mouse.OverrideCursor = null;

DXMessageBox.Show(error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

this.TeamRechHistorySearch.IsEnabled = true;

Mouse.OverrideCursor = null;

this.IsBusy = false;

}

}

#endregion

#region Private Metods

private byte[] imageOpenDialogBox()

{

System.Windows.Forms.OpenFileDialog fileOpenDialogBox = new System.Windows.Forms.OpenFileDialog();

fileOpenDialogBox.Filter = "JPEG|\*.jpg|BMP|\*.bmp|PNG|\*.png";

fileOpenDialogBox.Title = "Select a Images";

fileOpenDialogBox.FilterIndex = 1;

fileOpenDialogBox.RestoreDirectory = true;

byte[] imageInbytes = null;

if (fileOpenDialogBox.ShowDialog().Equals(System.Windows.Forms.DialogResult.OK))

{

FileInfo ImageInfo = new FileInfo(fileOpenDialogBox.FileName);

if (ImageInfo.Length < 819200)

{

imageInbytes = new MiraculousMethods().imageToByteArray(fileOpenDialogBox.FileName);

}

else

{

DXMessageBox.Show(CVsVariables.ERROR\_MESSAGES[0, 5], CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Stop);

}

}

return imageInbytes;

}

private void HideAllPanel()

{

var visiableGrids = this.LayoutRoot.Children.OfType<Grid>().Where(x => x.Visibility == Visibility.Visible);

foreach (Grid visiableGrid in visiableGrids)

{

visiableGrid.Visibility = Visibility.Hidden;

}

}

/// <summary>

/// Clear New Team Items

/// </summary>

private void NewTeamitemClear()

{

//this.newTeamImage.Source = null;

//this.newTeamTextBoxName.Text = string.Empty;

//this.newTeamComboBoxMemberName.Text = string.Empty;

//this.newMembers.Clear();

}

/// <summary>

/// Check Service connection null Or Not

/// </summary>

private void ServiceConnectionChecker()

{

if (ServerConnection.serviceFromServer == null || !ServerConnection.ServerAliveIs((ICommunicationObject)ServerConnection.serviceFromServer))

{

ServerConnection.ConnectToService();

}

}

private void RecharegHandelClear()

{

//this.RechHisButtonSearch.Click -= new RoutedEventHandler(RechHisButtonSearchClick);

//this.RechHisButtonSearch.Click -= new RoutedEventHandler(RechHisButtonSearchForTeamClick);

}

private void PasswordChangeClear()

{

this.settingChangeNewPassword.Password = string.Empty;

this.settingChangeOldPassword.Password = string.Empty;

this.settingChnageConPassword.Password = string.Empty;

}

#endregion

#region Propery Change

public event PropertyChangedEventHandler PropertyChanged;

private void OnPropertyChanged(string PropertyName)

{

if (PropertyChanged != null)

{

this.PropertyChanged(this, new PropertyChangedEventArgs(PropertyName));

}

}

#endregion

}

}

Class: StratWindow

namespace Procesta.CVClient

{

/// <summary>

/// Interaction logic for StratWindow.xaml

/// </summary>

public partial class StratWindow : Window, INotifyPropertyChanged

{

private BackgroundWorker startBackgroundWorker = new BackgroundWorker();

private string \_Messages;

public string Messages

{

get

{

return this.\_Messages;

}

set

{

this.\_Messages = value;

if (this.PropertyChanged != null)

{

this.PropertyChanged(this, new PropertyChangedEventArgs("Messages"));

}

}

}

/// <summary>

/// Window instillation

/// </summary>

public StratWindow()

{

this.InitializeComponent();

startBackgroundWorker.DoWork += new DoWorkEventHandler(startBackgroundWorker\_DoWork);

startBackgroundWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(startBackgroundWorker\_RunWorkerCompleted);

//if (!Properties.Settings.Default.IsConfigered)

//{

// new InstallWindow().Show();

// this.Close();

// return;

//}

startBackgroundWorker.RunWorkerAsync();

}

/// <summary>

///BackGroundWorker Work Complete

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

void startBackgroundWorker\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

if (e.Error == null)

{

new MainWindow().Show();

this.Close();

}

else

{

DXMessageBox.Show(e.Error.Message, CVsVariables.SOTWARE\_NAME, MessageBoxButton.OK, MessageBoxImage.Error);

this.Close();

}

}

/// <summary>

/// BackGroundWorker Do Work

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

void startBackgroundWorker\_DoWork(object sender, DoWorkEventArgs e)

{

try

{

this.Messages = "Checking...";

this.startupEntry();

Thread.Sleep(1000);

this.Messages = "Enabling Service";

ServerConnection.NotifactionToServer();

Thread.Sleep(1000);

this.Messages = "Enabling Command";

ServerConnection.NotifactionFromServer();

Thread.Sleep(1000);

this.Messages = "Checking to service";

ServerConnection.ConnectToService();

}

catch

{

throw;

}

}

/// <summary>

/// Registry Data Entry for startup

/// </summary>

private void startupEntry()

{

using (RegistryKey regKey = Registry.CurrentUser.OpenSubKey(@"Software\Microsoft\Windows\CurrentVersion\Run", true))

{

if (regKey != null)

{

object keyValue = regKey.GetValue("CvClient");

if (keyValue == null)

{

Registry.CurrentUser.OpenSubKey(@"Software\Microsoft\Windows\CurrentVersion\Run", true).SetValue("CVClient", System.Reflection.Assembly.GetEntryAssembly().Location, RegistryValueKind.String);

}

}

}

}

public event PropertyChangedEventHandler PropertyChanged;

}

}

**Class: TimeOut**

namespace Procesta.CVClient

{

/// <summary>

/// Interaction logic for TimeOut.xaml

/// </summary>

public partial class TimeOut : Window

{

[DllImport("user32.dll", EntryPoint = "FindWindow", SetLastError = true)]

static extern IntPtr FindWindow(string lpClassName, string lpWindowName);

[DllImport("user32.dll", EntryPoint = "SendMessage", SetLastError = true)]

static extern IntPtr SendMessage(IntPtr hWnd, Int32 Msg, IntPtr wParam, IntPtr lParam);

const int WM\_COMMAND = 0x111;

const int MIN\_ALL = 419;

const int MIN\_ALL\_UNDO = 416;

public TimeOut()

{

this.InitializeComponent();

IntPtr lHwnd = FindWindow("Shell\_TrayWnd", null);

SendMessage(lHwnd, WM\_COMMAND, (IntPtr)MIN\_ALL, IntPtr.Zero);

}

private void Storyboard\_Completed(object sender, EventArgs e)

{

new MainWindow().Show();

this.Close();

}

}

}