```
Note: User Permission

-CustomerView       : Allow User to view customer list

-CustomerCreate     : Allow user to create new customer

-CustomerModify     : Allow user to modify customer

-CustomerDelete     : Allow user to delete customer

-SaleView           : Allow user to view sale transaction

-SaleCreate         : Allow user to create sale transaction

-SaleModify         : Allow user to modify sale transaction

-SaleDelete         : Allow user to delete sale transaction
```

Database : Stored Procedure

```
Create Procedure spUserGet
(
    @userid    Int = 0
)
As
Begin
    If (@userid = 0)
       Begin
         Select * From [User];
       End
    Else
       Begin
         Select * From [User] Where UserId = @userid;
    End
End


Create Procedure spUserAdd
(
    @username             Varchar(50),
    @password             Varchar(50)
)
As
Begin
    Insert Into [User](Username,[Password],IsAdmin)
    Values(@username,@password,0);
End;


Create Procedure spUserUpdate
(
    @userid               Int,
    @username             Varchar(50),
    @password             Varchar(50)
)
As
Begin
    Update [User]
        Set
        Username = @username,
        [Password] = @password
    Where UserId = @userid;
End

Create Procedure spUserDelete
(
    @userid        Int
)
As
Begin
    Delete From [User] Where UserId = @userid;
End


Create Procedure spUserLogin
(
    @username             Varchar(50),
    @password             Varchar(50)
)
As
Begin
    Select * From AppUser Where Upper(Username) = Upper(@username) And [Password] =
@password;
End


Create Procedure spUserPermissionGet
(
    @userid    Int = 0
)
As
Begin
    If(@userid = 0)
      Begin
        Select * From UserPermission
      End
    Else
      Begin
        Select * From UserPermission Where UserId = @userid
    End
End


Create Procedure spUserPermissionAdd
(
    @userid                      Int,
    @permissionname       Varchar(50)
)
As
Begin
    Insert Into UserPermission(UserId,PermissionName)
    Values(@userid,@permissionname);
End


Create Procedure spUserPermissionDelete
(
    @userid      Int
)
As
Begin
    Delete From UserPermission Where UserId = @userid;
End
```

Data Layer : Models

```
public class User
    {
        public int UserId { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }
        public bool IsAdmin { get; set; }
    }
```

```
public class UserPermission
    {
        public int UserPermissionId { get; set; }
        public int UserId { get; set; }
        public string PermissionName { get; set; }
    }
```

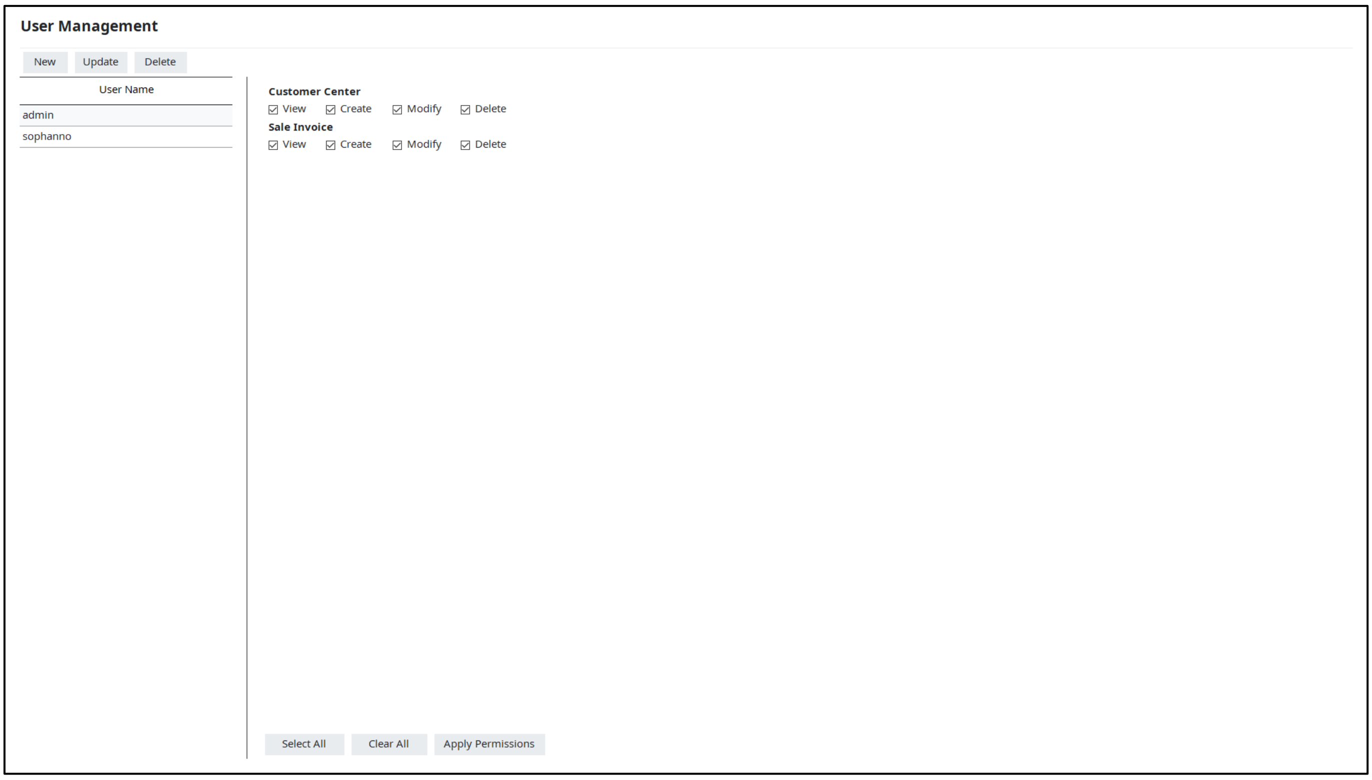
Data Layer : Data

```
public class UserService
    {
        public static DataTable GetAll()
        {
            SqlCommand command = new SqlCommand("spUserGet", POSContext.GetConnection());
            command.CommandType = CommandType.StoredProcedure;

            SqlDataAdapter adapter = new SqlDataAdapter(command);
            DataTable table = new DataTable();
            adapter.Fill(table);
            return table;
        }

        public static User Get(int userid)
        {
            User user = null;
            SqlCommand command = new SqlCommand("spUserGet", POSContext.GetConnection());
            command.CommandType = CommandType.StoredProcedure;

             command.Parameters.AddWithValue("@userid",userid);

            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                user = new User();
                user.UserId = Convert.ToInt32(reader["UserId"].ToString());
                user.Username = reader["UserName"].ToString();
                user.Password = reader["Password"].ToString();
                user.IsAdmin = Convert.ToBoolean(reader["IsAdmin"]);
            }
            return user;
        }

        public static void Add(User user)
        {
            try
            {
                SqlCommand command = new SqlCommand("spUserAdd",
POSContext.GetConnection());
                command.CommandType = CommandType.StoredProcedure;

                command.Parameters.AddWithValue("@username ",user.Username);
                command.Parameters.AddWithValue("@password ",user.Password);

                command.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        public static void Update(User user)
        {
            try
            {
                SqlCommand command = new SqlCommand("spUserUpdate",
POSContext.GetConnection());
                command.CommandType = CommandType.StoredProcedure;

                command.Parameters.AddWithValue("@userid ",user.UserId);
                command.Parameters.AddWithValue("@username ",user.Username);
                command.Parameters.AddWithValue("@password ",user.Password);

                command.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        public static void Delete(int userid)
        {
            try
            {
                SqlCommand command = new SqlCommand("spUserDelete",
POSContext.GetConnection());
                command.CommandType = CommandType.StoredProcedure;

                command.Parameters.AddWithValue("@userid", userid);

                command.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        public static User Login(string username, string password)
        {
            User user = null;
            SqlCommand command = new SqlCommand("spUserLogin", POSContext.GetConnection());
            command.CommandType = CommandType.StoredProcedure;

            command.Parameters.AddWithValue("@username", username);
            command.Parameters.AddWithValue("@password", password);

            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                user = new User();
                user.UserId = Convert.ToInt32(reader["UserId"].ToString());
                user.Username = reader["UserName"].ToString();
                user.Password = reader["Password"].ToString();
                user.IsAdmin = Convert.ToBoolean(reader["IsAdmin"]);
            }
            return user;
        }
    }
```

```
public class UserPermissionService
    {
        public static DataTable Get(int userid)
        {
            SqlCommand command = new SqlCommand("spUserPermissionGet",
POSContext.GetConnection());
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.AddWithValue("@userid",userid);

            SqlDataAdapter dapter = new SqlDataAdapter(command);
            DataTable table = new DataTable();
            dapter.Fill(table);
            return table;
        }

        public static void Add(UserPermission permission)
        {
            try
            {
                SqlCommand command = new SqlCommand("spUserPermissionAdd",
POSContext.GetConnection());
                command.CommandType = CommandType.StoredProcedure;

                command.Parameters.AddWithValue("@userid", permission.UserId);
                command.Parameters.AddWithValue("@permissionname",
permission.PermissionName);

                command.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        public static void Delete(int userid)
        {
            try
            {
                SqlCommand command = new SqlCommand("spUserPermissionDelete",
POSContext.GetConnection());
                command.CommandType = CommandType.StoredProcedure;

                command.Parameters.AddWithValue("@userid ",userid);

                command.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
    }
```

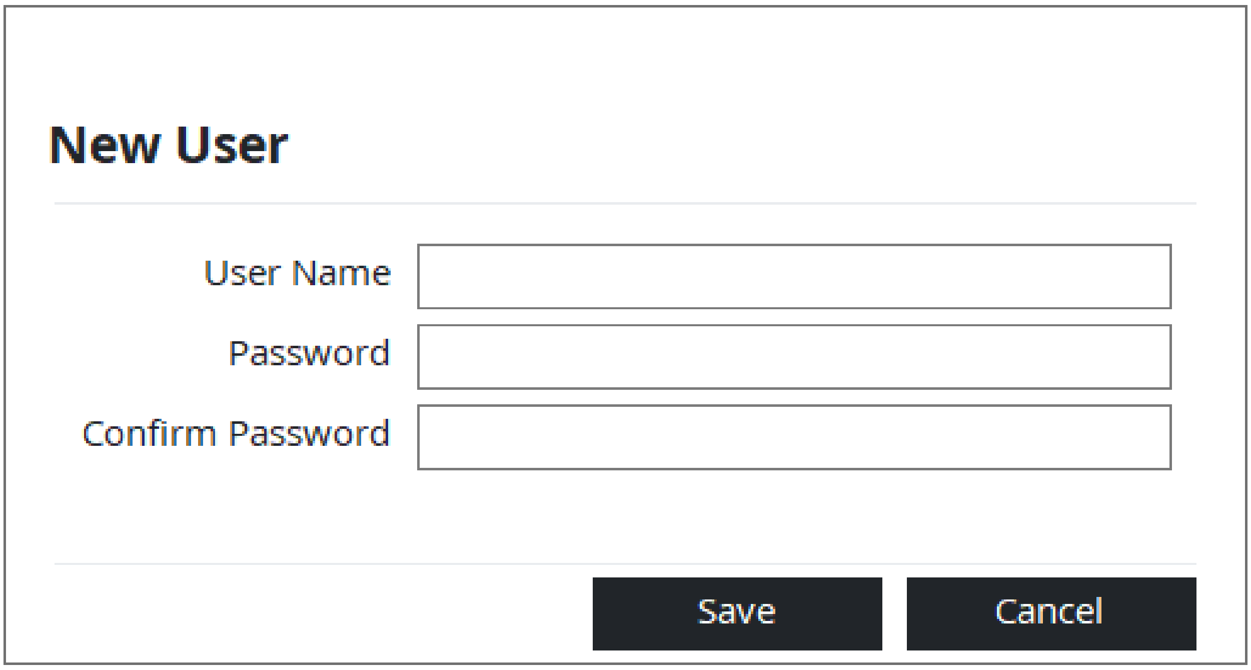
Business Layer : Forms

FormUserManagement



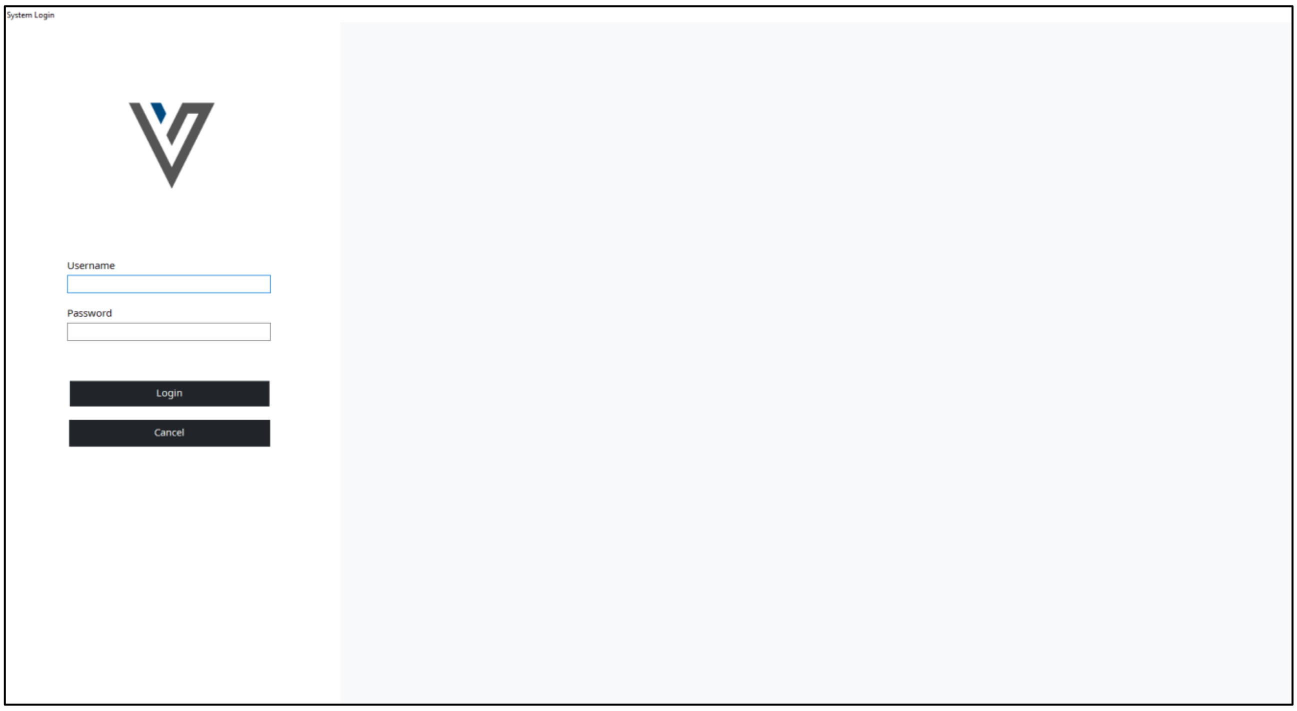
```
    public partial class FormUserManagement : Form
    {
        FormMain formMain;
        DataTable dtUser;
        DataTable dtUserPermission;
        public FormUserManagement(FormMain formMain)
        {
            this.formMain = formMain;
            InitializeComponent();
            InitializeData();
        }

        void InitializeData()
        {
            dtUser = UserService.GetAll();

            dgUsers.DataSource = dtUser;

            dgUsers.Columns[0].Visible = false;

            dgUsers.Columns[1].HeaderText = "User Name";
            dgUsers.Columns[1].Width = 200;
            dgUsers.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
            dgUsers.Columns[1].SortMode = DataGridViewColumnSortMode.NotSortable;
            dgUsers.Columns[1].Visible = true;

            dgUsers.Columns[2].Visible = false;

            dgUsers.Columns[3].Visible = false;
        }

        private void dgUsers_CellPainting(object sender, DataGridViewCellPaintingEventArgs
e)
        {
            if (e.RowIndex == -1 && e.ColumnIndex > -1)
            {
                e.Handled = true;
                using (Brush b = new SolidBrush(dgUsers.DefaultCellStyle.BackColor))
                {
                    e.Graphics.FillRectangle(b, e.CellBounds);
                }
                using (Pen p = new Pen(Brushes.Black))
                {
                    p.DashStyle = System.Drawing.Drawing2D.DashStyle.Solid;
                    p.Color = Color.FromArgb(33, 37, 41);
                    e.Graphics.DrawLine(p, new Point(0, e.CellBounds.Bottom - 1), new 
Point(e.CellBounds.Right, e.CellBounds.Bottom - 1));
                    e.Graphics.DrawLine(p, new Point(0, 0), new Point(e.CellBounds.Right, 
0));
                }
                e.PaintContent(e.ClipBounds);
            }
        }

        private void dgUsers_SelectionChanged(object sender, EventArgs e)
        {
            LoadUserPermission();
        }

        void LoadUserPermission()
        {
            if(dgUsers.SelectedRows.Count > 0)
            {
                int userid = 
Convert.ToInt32(dgUsers.SelectedRows[0].Cells["UserId"].Value.ToString());
                dtUserPermission = UserPermissionService.Get(userid);

                foreach(Control control in this.pnPermission.Controls)
                {
                    if(control is CheckBox)
                    {
                        ((CheckBox)control).Checked = false;
                    }
                }

                if(dtUserPermission.Rows.Count > 0)
                {
                    foreach(DataRow row in dtUserPermission.Rows)
                    {
                        CheckBox permission = 
(CheckBox)this.pnPermission.Controls[row["PermissionName"].ToString()];
                        if (permission != null)
                            permission.Checked = true;
                    }
                }
            }
        }

        private void btnSelectAll_Click(object sender, EventArgs e)
        {
            foreach (Control control in this.pnPermission.Controls)
            {
                if (control is CheckBox)
                {
                    ((CheckBox)control).Checked = true;
                }
            }
        }

        private void btnClearAll_Click(object sender, EventArgs e)
        {
            foreach (Control control in this.pnPermission.Controls)
            {
                if (control is CheckBox)
                {
                    ((CheckBox)control).Checked = false;
                }
            }
        }

        private void btnApplyPermission_Click(object sender, EventArgs e)
        {
            DialogResult confirmation;
            confirmation = MessageBox.Show("Do you really want to update user permission?", 
"Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (confirmation == DialogResult.Yes)
            {
                if (dgUsers.SelectedRows.Count > 0)
                {
                    int userid = 
Convert.ToInt32(dgUsers.SelectedRows[0].Cells["UserId"].Value.ToString());
                    AppUserPermissions.Delete(userid);
                    foreach (Control control in this.pnPermission.Controls)
                    {
                        if (control is CheckBox)
                        {
                            bool value = ((CheckBox)control).Checked;
                            if (value)
                            {
                                UserPermission permission = new UserPermission();
                                permission.UserId = userid;
                                permission.PermissionName = control.Name;
                                UserPermissionService.Add(permission);
                            }
                        }
                    }
                }
            }
        }

        private void btnNew_Click(object sender, EventArgs e)
        {
            FormUserManagementAddEdit formUserManagementAddEdit = new 
FormUserManagementAddEdit(null);
            if(formUserManagementAddEdit.ShowDialog() == DialogResult.OK)
            {
                InitializeData();
                dgUsers.Rows[dgUsers.Rows.Count - 1].Selected = true;
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            if (dgUsers.SelectedRows.Count > 0)
            {
                int selectedrow = dgUsers.SelectedRows[0].Index;
                int userid = 
Convert.ToInt32(dgUsers.SelectedRows[0].Cells["UserId"].Value.ToString());
                User user = UserService.Get(userid);
                if(user != null)
                {
                    FormUserManagementAddEdit formUserManagementAddEdit = new 
FormUserManagementAddEdit(user);
                    if (formUserManagementAddEdit.ShowDialog() == DialogResult.OK)
                    {
                        InitializeData();
                        dgUsers.Rows[selectedrow].Selected = true;
                    }
                }
            }
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            DialogResult confirmation;
            confirmation = MessageBox.Show("Do you really want to delete this user?",
"Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (confirmation == DialogResult.Yes)
            {
                if (dgUsers.SelectedRows.Count > 0)
                {
                    int userid =
Convert.ToInt32(dgUsers.SelectedRows[0].Cells["UserId"].Value.ToString());
                    UserService.Delete(userid);
                }
            }
        }
    }
```

FormUserManagementAddEdit



```
public partial class FormUserManagementAddEdit : Form
    {
        User _user;
        bool newuser;
        public FormUserManagementAddEdit(User user)
        {
            InitializeComponent();
            if(user == null)
            {
                this._user = new User();
                lblTitle.Text = "New User";
                this.newuser = true;
                txtUserName.Focus();
            }
            else
            {
                this._user = user;
                this.newuser = false;
                lblTitle.Text = "Edit User";
                InitializeData();
                txtUserName.Focus();
            }
        }

        void InitializeData()
        {
            txtUserName.Text = _user.Username;
            txtPassword.Text = _user.Password;
            txtConfirmPassword.Text = _user.Password;
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.Cancel;
            this.Close();
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            if (DoValidation())
            {
                _user.Username = txtUserName.Text.Trim();
                _user.Password = txtPassword.Text.Trim();
                if (newuser)
                    UserService.Add(_user);
                else
                    UserService.Update(_user);
                this.DialogResult = DialogResult.OK;
                this.Close();
            }
        }

        private bool DoValidation()
        {
            bool result = true;

            if(txtUserName.Text.Trim() == "")
            {
                epUserName.SetError(txtUserName, "Please enter User Name");
                result = false;
            }

            return result;
        }
    }
```

FormLogin



```
    public partial class FormLogin : Form
    {
        public FormLogin()
        {
            InitializeComponent();
        }

        private void btnLogin_Click(object sender, EventArgs e)
        {
            if (DoValidation())
            {
                User user =
UserService.Login(txtUsername.Text.Trim(),txtPassword.Text.Trim());
                if (user == null)
                {
                    MessageBox.Show("Invalid Username and Password, Please enter
again.","Information",MessageBoxButtons.OK,MessageBoxIcon.Information);
                    return;
                }
                FormMain formMain = new FormMain();
                //formMain.UserLogin = user;
                this.Hide();
                formMain.ShowDialog();
                this.Close();
            }
        }

        bool DoValidation()
        {
            bool result = true;
            if(txtUsername.Text.Trim() == "")
            {
                epUsername.SetError(txtUsername, "Please enter username.");
                result = false;
            }
            if(txtPassword.Text.Trim() == "")
            {
                epPassword.SetError(txtPassword, "Password cannot blank, Please enter
password");
                result = false;
            }

            return result;
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.Cancel;
            this.Close();
        }
    }
```