LAMPIRAN

FormLogin.pas

```
unit FormLogin;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Imaging.jpeg, Vcl.ExtCtrls,
  Vcl.Buttons, Vcl.StdCtrls, Data.DB, Data.Win.ADODB;

type
  TfrmLogin = class(TForm)
    GroupBox1: TGroupBox;
    edtUser: TEdit;
    edtPassword: TEdit;
    qrySQL: TADOQuery;
    Edit1: TEdit;
    Edit2: TEdit;
    Image1: TImage;
    btnLogin: TSpeedButton;
    btnExit: TSpeedButton;
    function CekValid : Boolean;
    function CekUser : Boolean;
    function CekPassword : Boolean;
    function CekLogin : Boolean;
    procedure OpenQrySQL;
    procedure Bersihkan;
    procedure btnLoginClick(Sender: TObject);
    procedure edtUserExit(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure btnExitClick(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    procedure edtUserKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure edtPasswordKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
```

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmLogin: TfrmLogin;
  temp, nama, role : String;

implementation
uses FormMenu, FormComponent, GlobalUnit;

{$R *.dfm}

procedure TfrmLogin.OpenQrySQL;
begin
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;
end;

procedure TfrmLogin.Bersihkan;
begin
  edtUser.Text    := '';
  edtPassword.Text:= '';
  nama            := '';
  role            := '';

  edtUser.SetFocus;
end;

procedure TfrmLogin.btnExitClick(Sender: TObject);
begin
  PostMessage(Self.Handle,wm_close,0,0);
end;
```

```
procedure TfrmLogin.btnLoginClick(Sender: TObject);
begin
  if CekValid then
  begin
    temp := 'SELECT * FROM [AdmUser] WHERE UserName = ' +
QuotedStr(edtUser.Text);
    OpenQrySQL;

    with qrySQL do
    begin
      Edit;
      FieldByName('LoginStat').AsString := 'Y';
      Post;
    end;

    frmComponent.lblUserName.Caption := edtUser.Text;

    with frmMenu do
    begin
      Label1.Caption := nama;
      Label2.Caption := '('+role+')';
      Label3.Caption := edtUser.Text;
      ShowModal;
    end;
  end;
end;

function TfrmLogin.CekValid : Boolean;
begin
  CekValid := True;

  if (StringReplace(edtUser.Text, ' ', '', [rfReplaceAll, rfIgnoreCase]) = '') then
  begin
    MessageDlg('User required!', mtError, [mbOK], 0);
    edtUser.SetFocus;
    CekValid := False;
  end
  else if (StringReplace(edtPassword.Text, ' ', '', [rfReplaceAll, rfIgnoreCase]) =
'') then
  begin
    MessageDlg('Password required!', mtError, [mbOK], 0);
    edtPassword.SetFocus;
    CekValid := False;
```

```
  end
  else if not CekUser then
  begin
    CekValid := False;
  end
  else if not CekPassword then
  begin
    CekValid := False;
  end
  else if not CekLogin then
  begin
    CekValid := False;
  end;
end;

procedure TfrmLogin.edtPasswordKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if (Key = VK_RETURN) then
  begin
    btnLogin.Click;
  end;
end;

procedure TfrmLogin.edtUserExit(Sender: TObject);
begin
  edtUser.Text := UpperCase(edtUser.Text);

  temp := 'SELECT * FROM [AdmUser] WHERE UserName = ' +
QuotedStr(edtUser.Text);
  OpenQrySQL;
  nama := qrySQL.FieldByName('Name').AsString;
  role := qrySQL.FieldByName('Role').AsString;
end;

procedure TfrmLogin.edtUserKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if (Key = VK_RETURN) then
  begin
    btnLogin.Click;
  end;
end;
```

```
procedure TfrmLogin.FormCloseQuery(Sender: TObject; var CanClose:
Boolean);
var userid : String;
begin
  userid := (StringReplace(edtUser.Text, 'User : ', '', [rfReplaceAll,
rfIgnoreCase]));
  if userid <> '' then
  begin
    temp := 'SELECT * FROM [AdmUser] WHERE UserName = ' +
QuotedStr(userid);
    OpenQrySQL;

    with qrySQL do
    begin
      if not Eof then
      begin
        Edit;
        FieldByName('LoginStat').AsString := 'N';
        Post;
      end;
    end;
    CanClose := True;
  end;
end;

procedure TfrmLogin.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if (Key = VK_RETURN) then
  begin
    btnLogin.Click;
  end;
end;

procedure TfrmLogin.FormShow(Sender: TObject);
begin
  Bersihkan;
end;
```

```
function TfrmLogin.CekUser : Boolean;
begin
  CekUser := True;
  temp := 'SELECT * FROM [AdmUser] WHERE UserName = ' +
QuotedStr(edtUser.Text);
  OpenQrySQL;

  if qrySQL.Eof then
  begin
    MessageDlg('User not found!', mtError, [mbOK], 0);
    edtUser.SetFocus;
    CekUser := False;
  end;
end;

function TfrmLogin.CekPassword : Boolean;
begin
  CekPassword := True;
  temp := 'SELECT * FROM [AdmUser] WHERE UserName = ' +
QuotedStr(edtUser.Text) ;
  OpenQrySQL;

  if (qrySQL.FieldByName('Password').AsString <> edtPassword.Text) then
  begin
    MessageDlg('Password incorrect!', mtError, [mbOK], 0);
    edtPassword.SetFocus;
    CekPassword := False;
  end;
end;

function TfrmLogin.CekLogin : Boolean;
begin
  CekLogin := True;
  if qrySQL.FieldByName('LoginStat').AsString = 'Y' then
  begin
    MessageDlg('You already login!', mtError, [mbOK], 0);
    CekLogin := False;
  end;
end;

end.
```

FormCreateJob.pas

```
unit FormCreateJob;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Buttons, Vcl.StdCtrls, Vcl.Mask,
  AdvDropDown, AdvCustomGridDropDown, AdvGridDropDown, Data.DB,
Data.Win.ADODB,
  Vcl.ComCtrls, StrUtils;

type
  TfrmCreateJob = class(TForm)
    GroupBox3: TGroupBox;
    Edit4: TEdit;
    edtJob: TEdit;
    Edit3: TEdit;
    Edit5: TEdit;
    edtStckDesc: TEdit;
    Edit6: TEdit;
    Edit2: TEdit;
    Edit7: TEdit;
    edtDtlQty: TEdit;
    Edit11: TEdit;
    Edit10: TEdit;
    cbStockCode: TAdvGridDropDown;
    edtJobDesc: TEdit;
    cbWarehouse: TAdvGridDropDown;
    qrySQL: TADOQuery;
    btnSave: TSpeedButton;
    cbUom: TAdvGridDropDown;
    dtpStart: TDateTimePicker;

    function CekSave : Boolean;

    procedure FormShow(Sender: TObject);
    procedure bersihkan;
    procedure isicombo;
    procedure OpenQrySQL;
    procedure cbWarehouseChange(Sender: TObject);
    procedure cbStockCodeChange(Sender: TObject);
```

```
    procedure btnSaveClick(Sender: TObject);
    procedure edtDtlQtyKeyPress(Sender: TObject; var Key: Char);
    procedure BuatCode;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmCreateJob: TfrmCreateJob;
  temp : String;
  tahun1, bulan1, hari : Word;

implementation
uses FormComponent, Numbering;

{$R *.dfm}

function TfrmCreateJob.CekSave : Boolean;
begin
  CekSave := True;

  if cbWarehouse.Text = '' then
  begin
    MessageDlg('Warehouse required!', mtError, [mbOK], 0);
    cbWarehouse.SetFocus;
    CekSave := False;
  end
  else
  if cbStockCode.Text = '' then
  begin
    MessageDlg('Stock Code required!', mtError, [mbOK], 0);
    cbStockCode.SetFocus;
    CekSave := False;
  end
  else
  if edtDtlQty.Text = '' then
  begin
    MessageDlg('Quantity to Make required!', mtError, [mbOK], 0);
    edtDtlQty.SetFocus;
    CekSave := False;
  end
```

```
  else
  if cbUom.Text = '' then
  begin
    MessageDlg('Unit of Measure required!', mtError, [mbOK], 0);
    cbUom.SetFocus;
    CekSave := False;
  end;
end;

procedure TfrmCreateJob.BuatCode;
var Panjang, NextSuffix : Integer;
    Prefix, flag, temp_left, bulan, tahun : String;
begin
  DecodeDate(dtpStart.Date, tahun1, bulan1, hari);
  bulan := IntToStr(bulan1);
  tahun := IntToStr(tahun1);

  if bulan1 = 1 then bulan := 'JAN';
  if bulan1 = 2 then bulan := 'FEB';
  if bulan1 = 3 then bulan := 'MAR';
  if bulan1 = 4 then bulan := 'APR';
  if bulan1 = 5 then bulan := 'MEI';
  if bulan1 = 6 then bulan := 'JUN';
  if bulan1 = 7 then bulan := 'JUL';
  if bulan1 = 8 then bulan := 'AGU';
  if bulan1 = 9 then bulan := 'SEP';
  if bulan1 = 10 then bulan := 'OKT';
  if bulan1 = 11 then bulan := 'NOV';
  if bulan1 = 12 then bulan := 'DES';

  temp := 'SELECT TOP 1 Job FROM [Pro_Job] ' +
          'WHERE DATEPART(MONTH, StartDate) = '+ IntToStr(bulan1)+
          ' AND DATEPART(YEAR, StartDate) = '+ QuotedStr(tahun)+
          ' ORDER By Job DESC';
  OpenQrySQL;

  if qrySQL.eof then
  begin
    NextSuffix := 1;
  end
  else
  begin
    qrySQL.Last;
```

```
    temp_left  := RightStr(qrySQL.FieldByName('Job').AsString, 4);
    NextSuffix := StrToInt(temp_left)+ 1;
  end;

  Prefix  := 'JOB/'+ bulan + tahun + '/';
  Panjang := 16;
  edtJob.Text := UpdateCode(IntToStr(NextSuffix),Prefix,Panjang);
end;

procedure TfrmCreateJob.edtDtlQtyKeyPress(Sender: TObject; var Key: Char);
begin
  if not (Key in [#8, '0'..'9', '.']) then
  begin
    Key := #0;
  end
  else if (Key = '.') and (Pos(Key, edtDtlQty.Text) > 0) then
  begin
    Key := #0;
  end;
end;

procedure TfrmCreateJob.btnSaveClick(Sender: TObject);
var QtyToMake : Double;
begin
  if CekSave then
  begin
    temp:= 'SELECT * FROM Pro_Job where Job = ' + QuotedStr(edtJob.Text);
    with qrySQL do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;

      if not frmComponent.adoFauzi.InTransaction then
      begin
        frmComponent.adoFauzi.BeginTrans;
        try
          QtyToMake :=
StrToFloat(StringReplace(edtDtlQty.Text,',',',[rfReplaceAll, rfIgnoreCase]));
```

```
        Insert;
        FieldByName('Job').AsString:= edtJob.Text;
        FieldByName('JobDescription').AsString:= edtJobDesc.Text;
        FieldByName('StockCode').AsString:= cbStockCode.Text;
        FieldByName('StockDescription').AsString:= edtStckDesc.Text;
        FieldByName('Warehouse').AsString:= cbWarehouse.Text;
        FieldByName('StartDate').AsDateTime:= dtpStart.Date;
        FieldByName('QtytoMake').AsFloat:= QtyToMake;
        FieldByName('QtyAccepted').AsFloat:= 0;
        FieldByName('QtyReceipt').AsFloat:= 0;
        FieldByName('QtyScrapped').AsFloat:= 0;
        FieldByName('Uom').AsString:= cbUom.Text;
        FieldByName('Status').AsString:= 'N';
        FieldByName('AddId').AsString:= frmComponent.lblUserName.Caption;
        Post;
        MessageDlg('Save success!',mtConfirmation,[mbOK],0);
        frmComponent.adoFauzi.CommitTrans;
        bersihkan;
        PostMessage(Self.Handle,wm_close,0,0);

        Except
        on E : Exception do
        begin
          frmComponent.adoFauzi.RollbackTrans;
          MessageDlg(E.Message, mtError, [mbOK], 0);
        end;
      end;
    end;
  end;
 end;
end;

procedure TfrmCreateJob.cbStockCodeChange(Sender: TObject);
begin
 temp := 'SELECT * FROM Inventory a, InventoryWh b '+
        ' where a.StockCode = b.StockCode and b.Warehouse = '+
        QuotedStr(cbWarehouse.Text) + ' and b.StockCode = '+
        QuotedStr(cbStockCode.Text);
 OpenQrySQL;

 cbUom.Clear;
 cbUom.Items.Clear;
 cbUom.ClearSelection;
```

```
  cbUom.Items.Add.Text.Add(qrySQL.FieldByName('Uom').AsString);
  cbUom.Items.Add.Text.Add(qrySQL.FieldByName('AltUom').AsString);
  cbUom.ItemIndex := -1;
  edtStckDesc.Text  := qrySQL.FieldByName('Description').AsString;
  edtDtlQty.Text    := '0';
end;

procedure TfrmCreateJob.cbWarehouseChange(Sender: TObject);
begin
  cbStockCode.Clear;
  cbStockCode.Items.Clear;
  cbStockCode.ClearSelection;
  cbUom.Clear;
  cbUom.Items.Clear;
  cbUom.ClearSelection;

  temp := 'SELECT a.StockCode, a.Description, b.QtyOnHand FROM Inventory
a, InventoryWh b '+
         ' where a.StockCode = b.StockCode and b.Warehouse = '+
         QuotedStr(cbWarehouse.Text) + ' and a.Category = ' +
         QuotedStr('M');
  OpenQrySQL;
  while not(qrySQL.Eof) do
  begin
    with cbStockCode.Items.Add do
    begin
      Text.Add(qrySQL.FieldByName('StockCode').AsString);
      Text.Add(qrySQL.FieldByName('Description').AsString);
      Text.Add(qrySQL.FieldByName('QtyOnHand').AsString);
    end;
    qrySQL.Next;
  end;
  cbStockCode.ItemIndex := -1;
  cbStockCode.Enabled   := true;
  edtStckDesc.Text := '';
  edtDtlQty.Text   := '0';
end;

procedure TfrmCreateJob.FormShow(Sender: TObject);
begin
  bersihkan;
  BuatCode;
end;
```

```
procedure TfrmCreateJob.bersihkan;
begin
  edtJob.Text := '';
  edtJobDesc.Text := '';
  edtStckDesc.Text := '';
  edtDtlQty.Text := '0';
  dtpStart.DateTime := Now;
  cbWarehouse.Clear;
  cbWarehouse.Items.Clear;
  cbWarehouse.ClearSelection;
  cbStockCode.Clear;
  cbStockCode.Items.Clear;
  cbStockCode.ClearSelection;
  cbUom.Clear;
  cbUom.Items.Clear;
  cbUom.ClearSelection;

  isicombo;
end;

procedure TfrmCreateJob.OpenQrySQL;
begin
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;
end;

procedure TfrmCreateJob.isicombo;
begin
  temp := 'SELECT DISTINCT a.Warehouse, b.Description FROM
[InventoryWh] a, '+
          '[Warehouse] b, [Inventory] c where a.Warehouse = b.Warehouse '+
          ' and a.StockCode = c.StockCode and c.Category = ' + QuotedStr('M')+
          ' and a.Warehouse <> ' +QuotedStr('WH004')+
          ' Order BY a.Warehouse ASC';
  OpenQrySQL;
  while not(qrySQL.Eof) do
  begin
```

```
    with cbWarehouse.Items.Add do
    begin
      Text.Add(qrySQL.FieldByName('Warehouse').AsString);
      Text.Add(qrySQL.FieldByName('Description').AsString);
    end;
    qrySQL.Next;
  end;

  cbWarehouse.ItemIndex := -1;
  cbStockCode.Enabled := False;
  cbUom.Clear;
  cbUom.Items.Clear;
  cbUom.ClearSelection;
end;

end.
```

FormJobIssue.pas

```
unit FormJobIssue;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Mask,
AdvDropDown,
  AdvCustomGridDropDown, AdvGridDropDown, Vcl.Grids, AdvObj,
BaseGrid, AdvGrid,
  DBAdvGrid, Data.DB, Data.Win.ADODB, Vcl.Buttons, Vcl.ComCtrls;

type
  TfrmJobIssue = class(TForm)
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    GroupBox4: TGroupBox;
    Edit4: TEdit;
    edtDtlJob: TEdit;
    Edit3: TEdit;
    edtDtlJobDesc: TEdit;
    Edit5: TEdit;
    edtDtlStckDesc: TEdit;
```

edtDtlStockCode: TEdit;
Edit6: TEdit;
Edit2: TEdit;
edtDtlWarehouse: TEdit;
Edit7: TEdit;
edtDtlQty: TEdit;
Edit11: TEdit;
edtDtlUom: TEdit;
Edit10: TEdit;
edtDtlDate: TEdit;
qryMaterial: TADOQuery;
dsMaterial: TDataSource;
qrySQL: TADOQuery;
btnPost: TSpeedButton;
TabelView: TDBAdvGrid;
qryMaterialJob: TWideStringField;
qryMaterialLine: TBCDField;
qryMaterialMStockCode: TWideStringField;
qryMaterialMDescription: TWideStringField;
qryMaterialMQtyIssued: TFMTBCDField;
qryMaterialMUom: TWideStringField;
qryMaterialMWarehouse: TWideStringField;
qryMaterialMUnitCost: TFMTBCDField;
qryMaterialTrnValue: TBCDField;
qryMaterialTrnDate: TDateTimeField;
qrySQL2: TADOQuery;
qryMaterialDescription: TWideStringField;
PageControl1: TPageControl;
Material: TTabSheet;
Edit12: TEdit;
cbWarehouse: TAdvGridDropDown;
Edit13: TEdit;
cbStockCode: TAdvGridDropDown;
Edit15: TEdit;
edtStckDesc: TEdit;
Edit17: TEdit;
edtQty: TEdit;
Edit21: TEdit;
cbUom: TAdvGridDropDown;
btnSubmit: TButton;
btnDelete: TButton;
Cost: TTabSheet;
Edit1: TEdit;

```
    edtCostDesc: TEdit;
    Edit9: TEdit;
    edtCost: TEdit;
    btnSubmitCost: TButton;
    btnDeleteCost: TButton;
    edtCostQty: TEdit;

    function CekSubmit : Boolean;
    function CekSubmitCost : Boolean;

    procedure FormShow(Sender: TObject);
    procedure bersihkan;
    procedure bacadata;
    procedure isicombo;
    procedure OpenQrySQL;
    procedure cbStockCodeChange(Sender: TObject);
    procedure cbWarehouseChange(Sender: TObject);
    procedure btnSubmitClick(Sender: TObject);
    procedure btnDeleteClick(Sender: TObject);
    procedure btnPostClick(Sender: TObject);
    procedure edtQtyExit(Sender: TObject);
    procedure edtQtyKeyPress(Sender: TObject; var Key: Char);
    procedure edtCostKeyPress(Sender: TObject; var Key: Char);
    procedure btnSubmitCostClick(Sender: TObject);
    procedure edtCostQtyKeyPress(Sender: TObject; var Key: Char);
    procedure PageControl1Change(Sender: TObject);
    procedure btnDeleteCostClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmJobIssue: TfrmJobIssue;
  temp : String;
  line, data : Integer;
  QtyonHand, QtyIssue : Double;

implementation
uses FormComponent, FormJobEntry;

{$R *.dfm}
```

```
procedure TfrmJobIssue.OpenQrySQL;
begin
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;
end;

procedure TfrmJobIssue.PageControl1Change(Sender: TObject);
begin
  bersihkan;
  if PageControl1.ActivePage = Material then
  begin
    isicombo;
    cbWarehouse.SetFocus;
  end
  else
    edtCostDesc.SetFocus;
end;

procedure TfrmJobIssue.btnPostClick(Sender: TObject);
begin
  if not qryMaterial.Eof then
  begin
    line := 0;
    temp := 'Select * From Pro_Issue Where Line = 1 and Job = ' +
          QuotedStr(edtDtlJob.Text);
    with qrySQL2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
    end;

    if qrySQL2.Eof then
    begin
      temp := 'Select * From Pro_Issue Where Line <> 1 and Job = ' +
            QuotedStr(edtDtlJob.Text);
      OpenQrySQL;
```

```
      with qrySQL do
      begin
        while not Eof do
        begin
          line := line + 1;
          Edit;
          FieldByName('Line').AsInteger := line;
          Post;
          Next;
        end;
      end;
    end;
    MessageDlg('Data Posted!',mtConfirmation,[mbOK],0);
    bersihkan;
    PostMessage(Self.Handle,wm_close,0,0);
  end
  else
  begin
    ShowMessage('There is an empty data!');
  end;
end;

procedure TfrmJobIssue.FormShow(Sender: TObject);
begin
  line := 0;
  data := 0;
  bacadata;
  bersihkan;
  isicombo;
  PageControl1.ActivePage := Material;
  cbWarehouse.SetFocus;
end;

procedure TfrmJobIssue.bersihkan;
begin
  cbWarehouse.Clear;
  cbWarehouse.Items.Clear;
  cbWarehouse.ClearSelection;
  cbStockCode.Clear;
  cbStockCode.Items.Clear;
  cbStockCode.ClearSelection;
```

```
  edtStckDesc.Text  := '';
  edtQty.Text       := '0';
  edtCostDesc.Text  := '';
  edtCostQty.Text   := '0';
  edtCost.Text      := '0';
  cbUom.Clear;
  cbUom.Items.Clear;
  cbUom.ClearSelection;

  with qryMaterial do
  begin
    Close;
    Parameters[0].Value := edtDtlJob.Text;
    Open;
  end;
end;

function TfrmJobIssue.CekSubmit : Boolean;
begin
  CekSubmit := True;

  if cbWarehouse.Text = '' then
  begin
    MessageDlg('Warehouse required!', mtError, [mbOK], 0);
    cbWarehouse.SetFocus;
    CekSubmit := False;
  end
  else
  if cbStockCode.Text = '' then
  begin
    MessageDlg('Stock Code required!', mtError, [mbOK], 0);
    cbStockCode.SetFocus;
    CekSubmit := False;
  end
  else
  if (edtQty.Text = '') or (edtQty.Text = '0') then
  begin
    MessageDlg('Quantity required!', mtError, [mbOK], 0);
    edtQty.SetFocus;
    CekSubmit := False;
  end
```

```
  else
  if cbUom.Text = '' then
  begin
    MessageDlg('UoM required!', mtError, [mbOK], 0);
    cbUom.SetFocus;
    CekSubmit := False;
  end
end;

function TfrmJobIssue.CekSubmitCost : Boolean;
begin
  CekSubmitCost := True;

  if edtCostDesc.Text = '' then
  begin
    MessageDlg('Description required!', mtError, [mbOK], 0);
    edtCostDesc.SetFocus;
    CekSubmitCost := False;
  end
  else
  if edtCost.Text = '' then
  begin
    MessageDlg('Cost required!', mtError, [mbOK], 0);
    edtCost.SetFocus;
    CekSubmitCost := False;
  end
  else
  if edtCostQty.Text = '0' then
  begin
    MessageDlg('Quantity required!', mtError, [mbOK], 0);
    edtCostQty.SetFocus;
    CekSubmitCost := False;
  end;
end;

procedure TfrmJobIssue.edtCostKeyPress(Sender: TObject; var Key: Char);
begin
  if not (Key in [#8, '0'..'9', '.']) then
  begin
    Key := #0;
  end
  else if (Key = '.') and (Pos(Key, edtCost.Text) > 0) then
  begin
```

```
    Key := #0;
  end;
end;

procedure TfrmJobIssue.edtCostQtyKeyPress(Sender: TObject; var Key: Char);
begin
  if not (Key in [#8, '0'..'9', '.']) then
  begin
    Key := #0;
  end
  else if (Key = '.') and (Pos(Key, edtCostQty.Text) > 0) then
  begin
    Key := #0;
  end;
end;

procedure TfrmJobIssue.edtQtyExit(Sender: TObject);
begin
  QtyIssue := StrToFloat(edtQty.Text);
  if QtyIssue > QtyonHand then
  begin
    edtQty.Text := '0';
    edtQty.SetFocus;
    if QtyonHand > 0 then
    begin
      ShowMessage('Max. '+ FloatToStr(QtyonHand));
    end
    else
      ShowMessage('Stock Empty!');
  end;
end;

procedure TfrmJobIssue.edtQtyKeyPress(Sender: TObject; var Key: Char);
begin
  if not (Key in [#8, '0'..'9', '.']) then
  begin
    Key := #0;
  end
  else if (Key = '.') and (Pos(Key, edtQty.Text) > 0) then
  begin
    Key := #0;
  end;
end;
```

```
procedure TfrmJobIssue.btnDeleteClick(Sender: TObject);
begin
  if not qryMaterial.Eof then
  begin
    if qryMaterial.FieldByName('MWarehouse').AsString <> 'NON' then
    begin
      temp := 'SELECT * From [InventoryWh] Where StockCode = ' +
            QuotedStr(qryMaterial.FieldByName('MStockCode').AsString) +
            ' and Warehouse = '+
            QuotedStr(qryMaterial.FieldByName('MWarehouse').AsString);
      with qrySQL2 do
      begin
        Close;
        SQL.Clear;
        SQL.Add(temp);
        Open;
      end;

      with qrySQL2 do
      begin
        Edit;
        FieldByName('QtyOnHand').AsFloat :=
FieldByName('QtyOnHand').AsFloat +
qryMaterial.FieldByName('MQtyIssued').AsFloat;
        FieldByName('QtyAllocated').AsFloat :=
FieldByName('QtyAllocated').AsFloat -
qryMaterial.FieldByName('MQtyIssued').AsFloat;
        Post;
      end;

      temp := 'DELETE FROM Pro_Issue WHERE Job = ' +
QuotedStr(qryMaterial.FieldByName('Job').AsString)+
            ' and Line = '+ IntToStr(qryMaterial.FieldByName('Line').AsInteger);
      with qrySQL2 do
      begin
        Close;
        SQL.Clear;
        SQL.Add(temp);
        ExecSQL;
      end;
      qryMaterial.Close;
      qryMaterial.Parameters[0].Value := edtDtlJob.Text;
      qryMaterial.Open;
```

```
    bersihkan;
   end
   else
    ShowMessage('Not Allowed!');
 end;
end;

procedure TfrmJobIssue.btnDeleteCostClick(Sender: TObject);
begin
 if not qryMaterial.Eof then
 begin
  if qryMaterial.FieldByName('MWarehouse').AsString = 'NON' then
  begin
   temp := 'DELETE FROM Pro_Issue WHERE Job = ' +
QuotedStr(qryMaterial.FieldByName('Job').AsString)+
          ' and Line = '+ IntToStr(qryMaterial.FieldByName('Line').AsInteger);
    with qrySQL2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      ExecSQL;
    end;

    qryMaterial.Close;
    qryMaterial.Parameters[0].Value := edtDtlJob.Text;
    qryMaterial.Open;

    bersihkan;
    isicombo;
   end
   else
    ShowMessage('Not Allowed!');
 end;
end;

procedure TfrmJobIssue.btnSubmitClick(Sender: TObject);
begin
 if CekSubmit then
 begin
  temp := 'SELECT * From [InventoryWh] Where StockCode = ' +
        QuotedStr(cbStockCode.Text) + ' and Warehouse = '+
        QuotedStr(cbWarehouse.Text);
```

```
    with qrySQL2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
    end;

    temp := 'SELECT TOP 1 * FROM Pro_Issue where Job = ' +
QuotedStr(edtDtlJob.Text)+
            'order by Job DESC';
    OpenQrySQL;

    if qrySQL.Eof then
    begin
      line := 1;
    end
    else
    begin
      line := qrySQL.FieldByName('Line').AsInteger + 1;
    end;

    with qrySQL do
    begin
      Insert;
      FieldByName('Job').AsString:=edtDtlJob.Text;
      FieldByName('Line').AsInteger:=line;
      FieldByName('MStockCode').AsString:=cbStockCode.Text;
      FieldByName('MDescription').AsString:=edtStckDesc.Text;
      FieldByName('MQtyIssued').AsFloat:=StrToFloat(edtQty.Text);
      FieldByName('MUom').AsString:=cbUom.Text;
      FieldByName('MWarehouse').AsString:=cbWarehouse.Text;

FieldByName('MUnitCost').AsFloat:=qrySQL2.FieldByName('UnitCost').AsFlo
at;

FieldByName('TrnValue').AsFloat:=qrySQL2.FieldByName('UnitCost').AsFloat
* StrToFloat(edtQty.Text);
      FieldByName('TrnDate').AsDateTime:=Now;
      FieldByName('AddId').AsString := frmComponent.lblUserName.Caption;
      Post;
    end;
```

```
    with qrySQL2 do
    begin
      Edit;
      FieldByName('QtyOnHand').AsFloat :=
FieldByName('QtyOnHand').AsFloat - StrToFloat(edtQty.Text);
      FieldByName('QtyAllocated').AsFloat :=
FieldByName('QtyAllocated').AsFloat + StrToFloat(edtQty.Text);
      Post;
    end;
    bersihkan;
  end;
end;

procedure TfrmJobIssue.btnSubmitCostClick(Sender: TObject);
begin
  if CekSubmitCost then
  begin
    temp := 'SELECT TOP 1 * FROM Pro_Issue where Job = ' +
QuotedStr(edtDtlJob.Text)+
          'order by Job DESC';
    OpenQrySQL;

    if qrySQL.Eof then
    begin
      line := 1;
    end
    else
    begin
      line := qrySQL.FieldByName('Line').AsInteger + 1;
    end;

    with qrySQL do
    begin
      Insert;
      FieldByName('Job').AsString:=edtDtlJob.Text;
      FieldByName('Line').AsInteger:=line;
      FieldByName('MStockCode').AsString:='-';
      FieldByName('MDescription').AsString:=edtCostDesc.Text;
      FieldByName('MQtyIssued').AsFloat:=StrToFloat(edtCostQty.Text);
      FieldByName('MUom').AsString:='-';
      FieldByName('MWarehouse').AsString:='NON';
      FieldByName('MUnitCost').AsFloat:=StrToFloat(edtCost.Text);
```

```
    FieldByName('TrnValue').AsFloat:=StrToFloat(edtCost.Text) *
StrToFloat(edtCostQty.Text);
    FieldByName('TrnDate').AsDateTime:=Now;
    FieldByName('AddId').AsString := frmComponent.lblUserName.Caption;
    Post;
   end;
   bersihkan;
   isicombo;
  end;
end;

procedure TfrmJobIssue.cbStockCodeChange(Sender: TObject);
begin
  temp := 'SELECT a.Description, b.* FROM Inventory a, InventoryWh b '+
        ' where a.StockCode = b.StockCode and b.Warehouse = '+
        QuotedStr(cbWarehouse.Text) + ' and b.StockCode = '+
        QuotedStr(cbStockCode.Text);
  OpenQrySQL;

  edtStckDesc.Text  := qrySQL.FieldByName('Description').AsString;
  QtyonHand         := qrySQL.FieldByName('QtyOnHand').AsFloat;
  edtQty.Text       := '0';


  temp := 'SELECT distinct a.Uom, a.AltUom FROM Inventory a, InventoryWh
b '+
        ' where a.StockCode = b.StockCode and b.Warehouse = '+
        QuotedStr(cbWarehouse.Text) + ' and b.StockCode = '+
        QuotedStr(cbStockCode.Text);
  OpenQrySQL;

  cbUom.Clear;
  cbUom.Items.Clear;
  cbUom.ClearSelection;
  cbUom.Items.Add.Text.Add(qrySQL.FieldByName('Uom').AsString);
  cbUom.Items.Add.Text.Add(qrySQL.FieldByName('AltUom').AsString);
  cbUom.ItemIndex := -1;
end;

procedure TfrmJobIssue.cbWarehouseChange(Sender: TObject);
begin
  cbStockCode.Clear;
  cbStockCode.Items.Clear;
```

```
  cbStockCode.ClearSelection;
  cbUom.Clear;
  cbUom.Items.Clear;
  cbUom.ClearSelection;

  if cbWarehouse.Text <> '' then
  begin
    temp := 'SELECT COUNT(*) as total From Pro_Issue Where Job = ' +
          QuotedStr(edtDtlJob.Text);
    OpenQrySQL;

    data := qrySQL.FieldByName('total').AsInteger;
  end;

  if data = 0 then
  begin
    temp := 'SELECT a.StockCode, a.Description, b.QtyOnHand FROM
Inventory a, InventoryWh b '+
          ' where a.StockCode = b.StockCode and b.Warehouse = '+
          QuotedStr(cbWarehouse.Text);
    OpenQrySQL;

    while not(qrySQL.Eof) do
    begin
      with cbStockCode.Items.Add do
      begin
        Text.Add(qrySQL.FieldByName('StockCode').AsString);
        Text.Add(qrySQL.FieldByName('Description').AsString);
        Text.Add(qrySQL.FieldByName('QtyOnHand').AsString);
      end;
      qrySQL.Next;
    end;
  end
  else
  begin
    temp := 'SELECT a.StockCode, a.Description, b.QtyOnHand, b.Warehouse '+
          'FROM Inventory a, InventoryWh b '+
          ' where a.StockCode = b.StockCode and b.Warehouse = '+
          QuotedStr(cbWarehouse.Text);
    OpenQrySQL;

    while not(qrySQL.Eof) do
    begin
```

```
    temp := 'SELECT * FROM Pro_Issue WHERE MStockCode = ' +
          QuotedStr(qrySQL.FieldByName('StockCode').AsString) +
          ' and MWarehouse = ' +
QuotedStr(qrySQL.FieldByName('Warehouse').AsString)+
          ' and Job = ' + QuotedStr(edtDtlJob.Text);
    with qrySQL2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
    end;

    if qrySQL2.eof then
    begin
      with cbStockCode.Items.Add do
      begin
        Text.Add(qrySQL.FieldByName('StockCode').AsString);
        Text.Add(qrySQL.FieldByName('Description').AsString);
        Text.Add(qrySQL.FieldByName('QtyOnHand').AsString);
      end;
    end;
    qrySQL.Next;
   end;
  end;

  cbStockCode.ItemIndex := -1;
  cbStockCode.Enabled   := true;
  edtStckDesc.Text:= '';
  edtQty.Text     := '0';
end;

procedure TfrmJobIssue.isicombo;
begin
  cbWarehouse.Clear;
  cbWarehouse.Items.Clear;

  temp := 'SELECT DISTINCT a.Warehouse, b.Description FROM
[InventoryWh] a, '+
        '[Warehouse] b, [Inventory] c where a.Warehouse = b.Warehouse '+
        ' and a.StockCode = c.StockCode'+
        ' and a.Warehouse <> ' +QuotedStr('WH004')+
        ' Order BY a.Warehouse ASC';
```

```
  OpenQrySQL;
  while not(qrySQL.Eof) do
  begin
    with cbWarehouse.Items.Add do
    begin
      Text.Add(qrySQL.FieldByName('Warehouse').AsString);
      Text.Add(qrySQL.FieldByName('Description').AsString);
    end;
    qrySQL.Next;
  end;

  cbWarehouse.ItemIndex := -1;
  cbStockCode.ClearSelection;
  cbStockCode.Enabled := False;
  edtStckDesc.Text  := '';
  edtQty.Text       := '0';
  cbUom.Clear;
  cbUom.Items.Clear;
  cbUom.ClearSelection;
end;

procedure TfrmJobIssue.bacadata;
begin
  with frmJobEntry do
  begin
    with qryJob do
    begin
      edtDtlJob.Text := FieldByName('Job').AsString;
      edtDtlJobDesc.Text := FieldByName('JobDescription').AsString;
      edtDtlStockCode.Text := FieldByName('StockCode').AsString;
      edtDtlStckDesc.Text := FieldByName('StockDescription').AsString;
      edtDtlWarehouse.Text := FieldByName('WHDesc').AsString;
      edtDtlQty.Text := FieldByName('QtyToMake').AsString;
      edtDtlUom.Text := FieldByName('Uom').AsString;
      edtDtlDate.Text := FieldByName('StartDate').AsString;
    end;
  end;
end;

end.
```

FormJobEntry.pas

```
unit FormJobEntry;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids, AdvObj, BaseGrid, AdvGrid,
  DBAdvGrid, Vcl.StdCtrls, Vcl.Buttons, Data.DB, Data.Win.ADODB,
frxClass,
  frxDBSet;

type
  TfrmJobEntry = class(TForm)
    GroupBox1: TGroupBox;
    qryJob: TADOQuery;
    dsJob: TDataSource;
    GroupBox2: TGroupBox;
    TabelView: TDBAdvGrid;
    dsJobDetail: TDataSource;
    qryJobIssue: TADOQuery;
    DBAdvGrid1: TDBAdvGrid;
    qryJobJob: TWideStringField;
    qryJobJobDescription: TWideStringField;
    qryJobStockCode: TWideStringField;
    qryJobStockDescription: TWideStringField;
    qryJobWarehouse: TWideStringField;
    qryJobStartDate: TDateTimeField;
    qryJobFinishDate: TDateTimeField;
    qryJobQtyToMake: TFMTBCDField;
    qryJobQtyScrapped: TFMTBCDField;
    qryJobUom: TStringField;
    qryJobStatus: TStringField;
    qryJobAddId: TWideStringField;
    qryJobIssueJob: TWideStringField;
    qryJobIssueLine: TBCDField;
    qryJobIssueMStockCode: TWideStringField;
    qryJobIssueMDescription: TWideStringField;
    qryJobIssueMQtyIssued: TFMTBCDField;
    qryJobIssueMUom: TWideStringField;
    qryJobIssueMWarehouse: TWideStringField;
    qryJobIssueMUnitCost: TFMTBCDField;
```

```
qryJobIssueTrnValue: TBCDField;
qryJobIssueTrnDate: TDateTimeField;
qryJobIssueDescription: TWideStringField;
qryJobWHDesc: TWideStringField;
GroupBox3: TGroupBox;
btnCreateJob: TSpeedButton;
btnJobIssue: TSpeedButton;
qryJobName: TWideStringField;
qryJobIssueName: TWideStringField;
qrySQL: TADOQuery;
qryJobQtyReceipt: TFMTBCDField;
qryJobQtyAccepted: TFMTBCDField;
PerintahKerja: TfrxReport;
btnRefresh: TSpeedButton;
btnPrint: TSpeedButton;
printSPK: TfrxDBDataset;
qryPrint: TADOQuery;
WideStringField1: TWideStringField;
WideStringField2: TWideStringField;
WideStringField3: TWideStringField;
WideStringField4: TWideStringField;
WideStringField5: TWideStringField;
DateTimeField1: TDateTimeField;
DateTimeField2: TDateTimeField;
FMTBCDField1: TFMTBCDField;
FMTBCDField2: TFMTBCDField;
StringField1: TStringField;
StringField2: TStringField;
WideStringField6: TWideStringField;
WideStringField7: TWideStringField;
WideStringField8: TWideStringField;
FMTBCDField3: TFMTBCDField;
FMTBCDField4: TFMTBCDField;
qryPrintLine: TBCDField;
qryPrintMStockCode: TWideStringField;
qryPrintMDescription: TWideStringField;
qryPrintMQtyIssued: TFMTBCDField;
qryPrintMUom: TWideStringField;
qryPrintMWarehouse: TWideStringField;
qryPrintMUnitCost: TFMTBCDField;
qryPrintTrnValue: TBCDField;
qryPrintTrnDate: TDateTimeField;
procedure FormShow(Sender: TObject);
```

```
    procedure bersihkan;
    procedure DBAdvGrid1ClickCell(Sender: TObject; ARow, ACol: Integer);
    procedure btnCreateJobClick(Sender: TObject);
    procedure btnJobIssueClick(Sender: TObject);
    procedure btnRefreshClick(Sender: TObject);
    procedure btnPrintClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmJobEntry: TfrmJobEntry;

implementation
uses FormComponent, FormCreateJob, FormJobIssue, FormMenu;

{$R *.dfm}

procedure TfrmJobEntry.btnCreateJobClick(Sender: TObject);
begin
  frmCreateJob.ShowModal;
  bersihkan;
end;

procedure TfrmJobEntry.btnJobIssueClick(Sender: TObject);
begin
  frmJobIssue.ShowModal;
  bersihkan;
end;

procedure TfrmJobEntry.btnPrintClick(Sender: TObject);
var
  Memo: TfrxMemoView;
  Component: TfrxComponent;
begin
  if (not qryJob.Eof) and (not qryJobIssue.Eof) then
  begin
    with qryPrint do
    begin
      Close;
      Parameters[0].Value := qryJob.FieldByName('Job').AsString;
```

```
    Open;
   end;
   PerintahKerja.ShowReport;
  end
  else
   ShowMessage('There is an empty data!');
end;

procedure TfrmJobEntry.btnRefreshClick(Sender: TObject);
begin
  bersihkan;
end;

procedure TfrmJobEntry.DBAdvGrid1ClickCell(Sender: TObject; ARow,
  ACol: Integer);
var role : String;
begin
  temp := 'SELECT Role FROM [AdmUser] WHERE UserName = ' +
        QuotedStr(frmMenu.Label3.Caption);
  with qrySQL do
  begin
   Close;
   SQL.Clear;
   SQL.Add(temp);
   Open;
  end;

  role := qrySQL.FieldByName('Role').AsString;

  temp := 'SELECT * FROM [Authority] WHERE Role = ' +
        QuotedStr(role);
  with qrySQL do
  begin
   Close;
   SQL.Clear;
   SQL.Add(temp);
   Open;
  end;

  with qryJobIssue do
  begin
   Close;
   Parameters[0].Value := qryJob.FieldByName('Job').AsString;
```

```
  Open;
 end;

 with qrySQL do
 begin
  if qryJob.FieldByName('Status').AsString = 'N' then
  begin
   if FieldByName('ValJobIssue').AsBoolean = true then
   begin
    btnJobIssue.Enabled := True;
   end
   else
    btnJobIssue.Enabled := false;
  end
  else
  begin
   btnJobIssue.Enabled := false;
  end;
 end;
end;

procedure TfrmJobEntry.FormShow(Sender: TObject);
var role : String;
begin
 btnCreateJob.Enabled    := False;
 btnJobIssue.Enabled     := False;

 temp := 'SELECT Role FROM [AdmUser] WHERE UserName = ' +
       QuotedStr(frmMenu.Label3.Caption);
 with qrySQL do
 begin
  Close;
  SQL.Clear;
  SQL.Add(temp);
  Open;
 end;

 role := qrySQL.FieldByName('Role').AsString;

 temp := 'SELECT * FROM [Authority] WHERE Role = ' +
       QuotedStr(role);
 with qrySQL do
 begin
```

```
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;

  with qrySQL do
  begin
    if FieldByName('ValCreateJob').AsBoolean = true then
      btnCreateJob.Enabled := True;
  end;
  bersihkan;
end;

procedure TfrmJobEntry.bersihkan;
begin
  qryJob.Close;
  qryJob.Open;
  qryJobIssue.Close;
end;

end.
```

FormJobReceipt.pas

```
unit FormJobReceipt;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Buttons, Data.DB,
  Data.Win.ADODB, Vcl.Grids, AdvObj, BaseGrid, AdvGrid, DBAdvGrid,
StrUtils;

type
  TfrmJobReceipt = class(TForm)
    GroupBox1: TGroupBox;
    Edit4: TEdit;
    edtJobDesc: TEdit;
    Edit15: TEdit;
    edtInfoWrh: TEdit;
    Edit12: TEdit;
```

edtInfoStockCode: TEdit;
Edit13: TEdit;
edtInfoStckDesc: TEdit;
Edit17: TEdit;
edtQtyOutst: TEdit;
Edit18: TEdit;
edtQtyToMake: TEdit;
Edit19: TEdit;
edtQtyManuf: TEdit;
Edit20: TEdit;
edtQtyScrpp: TEdit;
Edit25: TEdit;
edtStrtDate: TEdit;
btnPost: TSpeedButton;
GroupBox2: TGroupBox;
edtDtlUom: TEdit;
Edit11: TEdit;
Edit10: TEdit;
edtQty: TEdit;
Edit8: TEdit;
Edit9: TEdit;
ckJobComplete: TCheckBox;
Edit2: TEdit;
GroupBox3: TGroupBox;
qrySQL: TADOQuery;
cbJob: TComboBox;
qryMaterial: TADOQuery;
qryMaterialJob: TWideStringField;
qryMaterialLine: TBCDField;
qryMaterialMStockCode: TWideStringField;
qryMaterialMDescription: TWideStringField;
qryMaterialMQtyIssued: TFMTBCDField;
qryMaterialMUom: TWideStringField;
qryMaterialMWarehouse: TWideStringField;
qryMaterialMUnitCost: TFMTBCDField;
qryMaterialTrnValue: TBCDField;
qryMaterialTrnDate: TDateTimeField;
dsMaterial: TDataSource;
TabelView: TDBAdvGrid;
qrySQL2: TADOQuery;
qryMaterialWHDesc: TWideStringField;
Edit1: TEdit;
edtQtyAcc: TEdit;

```
    function CekSave : Boolean;
    procedure FormShow(Sender: TObject);
    procedure bersihkan;
    procedure cbJobChange(Sender: TObject);
    procedure edtQtyKeyPress(Sender: TObject; var Key: Char);
    procedure btnPostClick(Sender: TObject);
    procedure BuatCode;
    procedure edtQtyExit(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmJobReceipt: TfrmJobReceipt;
  temp, InspectCode : string;
  tahun1, bulan1, hari : Word;
  QtyReceipt, QtyToMake : Double;

implementation
uses FormComponent, Numbering;

{$R *.dfm}

function TfrmJobReceipt.CekSave : Boolean;
begin
  CekSave := True;

  if cbJob.Text = '' then
  begin
    MessageDlg('Job required!', mtError, [mbOK], 0);
    cbJob.SetFocus;
    CekSave := False;
  end
  else
  if (edtQty.Text = '') or (edtQty.Text = '0') then
  begin
    MessageDlg('Quantity required!', mtError, [mbOK], 0);
    edtQty.SetFocus;
    CekSave := False;
  end;
end;
```

```
procedure TfrmJobReceipt.btnPostClick(Sender: TObject);
var Qty, UnitCost : Double;
begin
  if CekSave then
  begin
    BuatCode;

    if(ckJobComplete.Checked=True) then
    begin
      while not qryMaterial.Eof do
      begin
        if qryMaterial.FieldByName('MWarehouse').AsString <> 'NON' then
        begin
          temp := 'SELECT * FROM [InventoryWh] Where StockCode = '+
                  QuotedStr(qryMaterial.FieldByName('MStockCode').AsString)+
                  ' and Warehouse = '+
                  QuotedStr(qryMaterial.FieldByName('MWarehouse').AsString);
          with qrySQL2 do
          begin
            Close;
            SQL.Clear;
            SQL.Add(temp);
            Open;

            Edit;
            FieldByName('QtyAllocated').AsFloat :=
FieldByName('QtyAllocated').AsFloat -
qryMaterial.FieldByName('MQtyIssued').AsFloat;
            Post;
          end;
        end;
        qryMaterial.Next;
      end;
    end;

    temp:= 'SELECT * FROM Pro_Job where Job = ' + QuotedStr(cbJob.Text);
    with qrySQL do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
```

```
    if not frmComponent.adoFauzi.InTransaction then
    begin
      frmComponent.adoFauzi.BeginTrans;
      try
        Qty := StrToFloat(StringReplace(edtQty.Text,',',",[rfReplaceAll,
rfIgnoreCase]));
        Edit;

FieldByName('QtyReceipt').AsFloat:=FieldByName('QtyReceipt').AsFloat+Qty;
        if(ckJobComplete.Checked=True) then
        begin
          FieldByName('Status').AsString:='Y';
        end
        else
        begin
          FieldByName('Status').AsString:='P';
        end;
        Post;

        temp:= 'SELECT sum(TrnValue) as Total FROM Pro_Issue where Job = '
+ QuotedStr(cbJob.Text);
        with qrySQL2 do
        begin
          Close;
          SQL.Clear;
          SQL.Add(temp);
          Open;

          UnitCost := FieldByName('Total').AsFloat/Qty;
        end;

        temp:= 'SELECT * FROM Pro_Inspect where InspectCode = ' +
QuotedStr(InspectCode);
        with qrySQL2 do
        begin
          Close;
          SQL.Clear;
          SQL.Add(temp);
          Open;
        end;

        qrySQL2.Insert;
        qrySQL2.FieldByName('InspectCode').AsString:= InspectCode;
```

```
        qrySQL2.FieldByName('Job').AsString:=cbJob.Text;
        qrySQL2.FieldByName('JobDescription').AsString:=edtJobDesc.Text;
        qrySQL2.FieldByName('StockCode').AsString:=edtInfoStockCode.Text;
        qrySQL2.FieldByName('Warehouse').AsString:=edtInfoWrh.Text;
        qrySQL2.FieldByName('ReceiptDate').AsDateTime:=Now;
        qrySQL2.FieldByName('QtyReceipt').AsFloat:=Qty;
        qrySQL2.FieldByName('QtyInspected').AsFloat:=0;
        qrySQL2.FieldByName('QtyAccepted').AsFloat:=0;
        qrySQL2.FieldByName('QtyScrapped').AsFloat:=0;
        qrySQL2.FieldByName('InspectCompleted').AsString:='N';
        qrySQL2.FieldByName('UnitCost').AsFloat:=UnitCost;
        qrySQL2.FieldByName('OriginalValue').AsFloat:=UnitCost*Qty;
        qrySQL2.FieldByName('CostUom').AsString:='';
        qrySQL2.FieldByName('Uom').AsString:=edtDtlUom.Text;
        qrySQL2.FieldByName('AddReceiptId').AsString :=
frmComponent.lblUserName.Caption;
        qrySQL2.Post;

        MessageDlg('Data Posted!',mtConfirmation,[mbOK],0);
        frmComponent.adoFauzi.CommitTrans;
        bersihkan;
        PostMessage(Self.Handle,wm_close,0,0);

        Except
        on E : Exception do
        begin
          frmComponent.adoFauzi.RollbackTrans;
          MessageDlg(E.Message, mtError, [mbOK], 0);
        end;
      end;
    end;
  end;
 end;
end;

procedure TfrmJobReceipt.BuatCode;
var Panjang, NextSuffix : Integer;
   Prefix, flag, temp_left, bulan, tahun : String;
begin
 DecodeDate(Date(), tahun1, bulan1, hari);
 bulan := IntToStr(bulan1);
 tahun := IntToStr(tahun1);
```

```
  if bulan1 = 1 then bulan := 'JAN';
  if bulan1 = 2 then bulan := 'FEB';
  if bulan1 = 3 then bulan := 'MAR';
  if bulan1 = 4 then bulan := 'APR';
  if bulan1 = 5 then bulan := 'MEI';
  if bulan1 = 6 then bulan := 'JUN';
  if bulan1 = 7 then bulan := 'JUL';
  if bulan1 = 8 then bulan := 'AGU';
  if bulan1 = 9 then bulan := 'SEP';
  if bulan1 = 10 then  bulan := 'OKT';
  if bulan1 = 11 then bulan := 'NOV';
  if bulan1 = 12 then bulan := 'DES';

  temp := 'SELECT TOP 1 InspectCode FROM [Pro_Inspect] ' +
        'WHERE DATEPART(MONTH, ReceiptDate) = '+ IntToStr(bulan1)+
        ' AND DATEPART(YEAR, ReceiptDate) = '+ QuotedStr(tahun)+
        ' ORDER By InspectCode DESC';
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;

  if qrySQL.eof then
  begin
    NextSuffix := 1;
  end
  else
  begin
    qrySQL.Last;
    temp_left  := RightStr(qrySQL.FieldByName('InspectCode').AsString, 4);
    NextSuffix := StrToInt(temp_left)+ 1;
  end;

  Prefix  := 'INS/'+ bulan + tahun + '/';
  Panjang := 16;
  InspectCode := UpdateCode(IntToStr(NextSuffix),Prefix,Panjang);
end;
```

```
procedure TfrmJobReceipt.cbJobChange(Sender: TObject);
begin
  temp := 'SELECT * FROM [Pro_Job] Where Job = ' +
          QuotedStr(cbJob.Text);
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
    edtJobDesc.Text := FieldByName('JobDescription').AsString;
    edtDtlUom.Text := FieldByName('Uom').AsString;
    edtInfoStockCode.Text := FieldByName('StockCode').AsString;
    edtInfoStckDesc.Text := FieldByName('StockDescription').AsString;
    edtInfoWrh.Text := FieldByName('Warehouse').AsString;

    if (FieldByName('QtyToMake').AsFloat)=0 then
    begin
      edtQtyOutst.Text:=FloatToStrF(0,ffNumber, 18, 2);
    end
    else
    begin
      edtQtyOutst.Text:=FloatToStrF((FieldByName('QtyToMake').AsFloat)-
(FieldByName('QtyReceipt').AsFloat),ffNumber, 18, 2);
    end;
edtQtyToMake.Text:=FloatToStrF(FieldByName('QtyToMake').AsFloat,ffNum
ber, 18, 2);
edtQtyManuf.Text:=FloatToStrF(FieldByName('QtyReceipt').AsFloat,ffNumber
, 18, 2);
edtQtyAcc.Text:=FloatToStrF(FieldByName('QtyAccepted').AsFloat,ffNumber,
18, 2);
edtQtyScrpp.Text:=FloatToStrF(FieldByName('QtyScrapped').AsFloat,ffNumbe
r, 18, 2);
    edtStrtDate.Text:=FieldByName('StartDate').AsString;
  end;

  with qryMaterial do
  begin
    Close;
    Parameters[0].Value := cbJob.Text;
    Open;
  end;
end;
```

```
procedure TfrmJobReceipt.edtQtyExit(Sender: TObject);
begin
  if cbJob.Text <> '' then
  begin
    QtyReceipt  := StrToFloat(edtQty.Text);
    QtyToMake   := StrToFloat(edtQtyOutst.Text);
    if QtyReceipt > QtyToMake then
    begin
      edtQty.Text := '0';
      edtQty.SetFocus;
      ShowMessage('Max. '+ FloatToStr(QtyToMake));
    end;
  end;
end;

procedure TfrmJobReceipt.edtQtyKeyPress(Sender: TObject; var Key: Char);
begin
  if not (Key in [#8, '0'..'9', '.']) then
  begin
    Key := #0;
  end
  else if (Key = '.') and (Pos(Key, edtQty.Text) > 0) then
  begin
    Key := #0;
  end;
end;

procedure TfrmJobReceipt.FormShow(Sender: TObject);
begin
  bersihkan;
end;

procedure TfrmJobReceipt.bersihkan;
begin
  edtJobDesc.Text := '';
  edtQty.Text := '0';
  ckJobComplete.Enabled := true;
  ckJobComplete.Checked := false;
  edtDtlUom.Text := '';
  edtInfoStockCode.Text := '';
  edtInfoStckDesc.Text := '';
  edtInfoWrh.Text := '';
  edtQtyOutst.Text := '';
```

```
  edtQtyToMake.Text := '';
  edtQtyManuf.Text := '';
  edtQtyScrpp.Text := '';
  edtStrtDate.Text := '';
  cbJob.Clear;
  cbJob.Items.Clear;
  cbJob.ClearSelection;

  temp := 'SELECT Job FROM [Pro_Job] Where Status <> ' +
        QuotedStr('Y');
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;

  while not qrySQL.Eof do
  begin
    cbJob.Items.Add(qrySQL.FieldByName('Job').AsString);
    qrySQL.Next;
  end;
  cbJob.ItemIndex := -1;
  qryMaterial.Close;
  qryMaterial.Open;
end;

end.
```

FormViewReceiptJob.pas

```
unit FormViewReceiptJob;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Buttons, Vcl.Grids, AdvObj,
  BaseGrid, AdvGrid, DBAdvGrid, Vcl.StdCtrls, Data.DB, Data.Win.ADODB;

type
  TfrmViewReceiptJob = class(TForm)
```

```
    GroupBox2: TGroupBox;
    DBAdvGrid2: TDBAdvGrid;
    qryReceipt: TADOQuery;
    dsReceipt: TDataSource;
    qryReceiptInspectCode: TWideStringField;
    qryReceiptJob: TWideStringField;
    qryReceiptJobDescription: TWideStringField;
    qryReceiptStockCode: TWideStringField;
    qryReceiptWarehouse: TWideStringField;
    qryReceiptReceiptDate: TDateTimeField;
    qryReceiptQtyInspected: TFMTBCDField;
    qryReceiptQtyAccepted: TFMTBCDField;
    qryReceiptQtyScrapped: TFMTBCDField;
    qryReceiptInspectCompleted: TStringField;
    qryReceiptUnitCost: TFMTBCDField;
    qryReceiptOriginalValue: TBCDField;
    qryReceiptCostUom: TWideStringField;
    qryReceiptUom: TWideStringField;
    qryReceiptType: TStringField;
    qryReceiptTimeStamp: TBytesField;
    qryReceiptWHDesc: TWideStringField;
    qryReceiptStockDesc: TWideStringField;
    GroupBox1: TGroupBox;
    btnReceipt: TSpeedButton;
    qryReceiptName: TWideStringField;
    qrySQL: TADOQuery;
    qryReceiptQtyReceipt: TFMTBCDField;
    procedure btnReceiptClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmViewReceiptJob: TfrmViewReceiptJob;

implementation
uses FormCOmponent, FormJobReceipt, FormMenu;

{$R *.dfm}
```

```
procedure TfrmViewReceiptJob.btnReceiptClick(Sender: TObject);
begin
  frmJobReceipt.ShowModal;
  qryReceipt.Close;
  qryReceipt.Open;
end;

procedure TfrmViewReceiptJob.FormShow(Sender: TObject);
var role : String;
begin
  btnReceipt.Enabled     := False;

  temp := 'SELECT Role FROM [AdmUser] WHERE UserName = ' +
        QuotedStr(frmMenu.Label3.Caption);
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;

  role := qrySQL.FieldByName('Role').AsString;

  temp := 'SELECT * FROM [Authority] WHERE Role = ' +
        QuotedStr(role);
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;

  with qrySQL do
  begin
    if FieldByName('ValViewJobReceipt').AsBoolean = true then
      btnReceipt.Enabled := True;
  end;
  qryReceipt.Close;
  qryReceipt.Open;
end;
end.
```

FormJobInspect.pas

```
unit FormJobInspect;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Buttons, Data.DB,
  Data.Win.ADODB;

type
  TfrmJobInspect = class(TForm)
    GroupBox2: TGroupBox;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Edit7: TEdit;
    Edit8: TEdit;
    edtDtlJob: TEdit;
    edtDtlJobDesc: TEdit;
    edtDtlStockCode: TEdit;
    edtDtlStockDesc: TEdit;
    edtDtlWh: TEdit;
    edtDtlWhDesc: TEdit;
    Edit9: TEdit;
    Edit10: TEdit;
    Edit11: TEdit;
    Edit12: TEdit;
    Edit13: TEdit;
    edtQtyAcc: TEdit;
    edtQtyScrapped: TEdit;
    edtComplete: TEdit;
    edtQtyInsp: TEdit;
    edtQtyManuf: TEdit;
    GroupBox1: TGroupBox;
    btnPost: TSpeedButton;
    btnScrap: TSpeedButton;
    btnAccept: TSpeedButton;
    Edit14: TEdit;
    Edit15: TEdit;
    Edit16: TEdit;
```

```
    edtInsUom: TEdit;
    edtInsQty: TEdit;
    Edit17: TEdit;
    ckInsComplete: TCheckBox;
    Edit2: TEdit;
    cbInspect: TComboBox;
    qrySQL: TADOQuery;
    qrySQL2: TADOQuery;
    Label2: TLabel;

    function CekSave : Boolean;
    procedure btnAcceptClick(Sender: TObject);
    procedure btnScrapClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure bersihkan;
    procedure cbInspectChange(Sender: TObject);
    procedure edtInsQtyKeyPress(Sender: TObject; var Key: Char);
    procedure btnPostClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure edtInsQtyExit(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmJobInspect: TfrmJobInspect;
  qtyInspect, qtyMax : Double;

implementation
uses FormComponent, FormCreateSerialNumber, FormInputDataScrap,
FormMenu;

{$R *.dfm}

procedure TfrmJobInspect.btnAcceptClick(Sender: TObject);
begin
  frmCreateSerialNumber.ShowModal;
end;
```

```
function TfrmJobInspect.CekSave : Boolean;
begin
  CekSave := True;

  if cbInspect.Text = '' then
  begin
    MessageDlg('Inspect Code required!', mtError, [mbOK], 0);
    cbInspect.SetFocus;
    CekSave := False;
  end
  else
  if (edtInsQty.Text = '') or (edtInsQty.Text = '0') then
  begin
    MessageDlg('Quantity Inspected required!', mtError, [mbOK], 0);
    edtInsQty.SetFocus;
    CekSave := False;
  end;
end;

procedure TfrmJobInspect.btnPostClick(Sender: TObject);
var Qty, QtyTemp, QtyOnHand,
    UnitCostInspect, UnitCostAsli, UnitCostBaru : Double;
begin
  if CekSave then
  begin
    temp := 'SELECT * FROM [Pro_Inspect] Where InspectCode = ' +
          QuotedStr(cbInspect.Text);
    with qrySQL do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;

      if not frmComponent.adoFauzi.InTransaction then
      begin
        frmComponent.adoFauzi.BeginTrans;
        try
          Qty := StrToFloat(StringReplace(edtInsQty.Text,',','',[rfReplaceAll,
rfIgnoreCase]));
          QtyTemp := FieldByName('QtyInspected').AsFloat;
          UnitCostInspect:= FieldByName('UnitCost').AsFloat;
```

```
        Edit;
        FieldByName('QtyInspected').AsFloat:= QtyTemp+Qty;
        FieldByName('AddInspectId').AsString :=
frmComponent.lblUserName.Caption;

        if QtyTemp = StrToFloat(edtQtyManuf.Text) then
        begin
          FieldByName('InspectCompleted').AsString := 'Y';
        end
        else
          FieldByName('InspectCompleted').AsString := 'N';
        Post;

        temp := 'SELECT * FROM [InventoryWh] Where StockCode = '+
              QuotedStr(frmJobInspect.edtDtlStockCode.Text)+
              ' and Warehouse = '+
              QuotedStr(frmJobInspect.edtDtlWh.Text);
        with qrySQL2 do
        begin
          Close;
          SQL.Clear;
          SQL.Add(temp);
          Open;

          QtyOnHand := FieldByName('QtyOnHand').AsFloat;
          UnitCostAsli := FieldByName('UnitCost').AsFloat;
          UnitCostBaru :=
((Qty*UnitCostInspect)+(QtyOnHand*UnitCostAsli))/(Qty+QtyOnHand);

          Edit;
          FieldByName('QtyInInspection').AsFloat:=
FieldByName('QtyInInspection').AsFloat+Qty;
          FieldByName('QtyOnHand').AsFloat:=
FieldByName('QtyOnHand').AsFloat+Qty;
          FieldByName('UnitCost').AsFloat:= UnitCostBaru;
          Post;
        end;

        Label2.Caption := edtInsQty.Text;
        edtInsQty.Enabled := false;
        MessageDlg('Data Posted!',mtConfirmation,[mbOK],0);
        frmComponent.adoFauzi.CommitTrans;
        cbInspectChange(self);
```

```
        Except
        on E : Exception do
        begin
          frmComponent.adoFauzi.RollbackTrans;
          MessageDlg(E.Message, mtError, [mbOK], 0);
        end;
      end;
    end;
  end;
end;

procedure TfrmJobInspect.btnScrapClick(Sender: TObject);
begin
  frmInputDataScrap.ShowModal;
end;

procedure TfrmJobInspect.cbInspectChange(Sender: TObject);
begin
  temp := 'SELECT a.*, b.Description FROM [Pro_Inspect] a, [Inventory] b '+
        'Where a.InspectCode = ' + QuotedStr(cbInspect.Text) +
        ' And a.StockCode = b.StockCode';
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;

  edtDtlJob.Text:=qrySQL.FieldByName('Job').AsString;
  edtDtlJobDesc.Text:=qrySQL.FieldByName('JobDescription').AsString;
  edtDtlStockCode.Text:=qrySQL.FieldByName('StockCode').AsString;
  edtDtlStockDesc.Text:=qrySQL.FieldByName('Description').AsString;
  edtDtlWh.Text:=qrySQL.FieldByName('Warehouse').AsString;
  edtDtlWhDesc.Text:=qrySQL.FieldByName('Description').AsString;
edtQtyManuf.Text:=FloatToStrF(qrySQL.FieldByName('QtyReceipt').AsFloat,f
ffFixed, 8,2);
edtQtyInsp.Text:=FloatToStrF(qrySQL.FieldByName('QtyInspected').AsFloat,ff
Fixed, 8,2);
edtQtyScrapped.Text:=FloatToStrF(qrySQL.FieldByName('QtyScrapped').AsFl
oat,ffFixed, 8,2);
```

```
edtQtyAcc.Text:=FloatToStrF(qrySQL.FieldByName('QtyAccepted').AsFloat,ff
Fixed, 8,2);
  edtInsUom.Text:=qrySQL.FieldByName('Uom').AsString;
end;

procedure TfrmJobInspect.edtInsQtyExit(Sender: TObject);
begin
  if cbInspect.Text <> '' then
  begin
    qtyInspect := StrToFloat(edtInsQty.Text);
    qtyMax     := StrToFloat(edtQtyManuf.Text)-StrToFloat(edtQtyInsp.Text);

 if qtyInspect > qtyMax then
    begin
      edtInsQty.Text := '0';
      edtInsQty.SetFocus;
      ShowMessage('Max. '+ FloatToStr(qtyMax));
    end;
  end;
end;

procedure TfrmJobInspect.edtInsQtyKeyPress(Sender: TObject; var Key: Char);
begin
  if not (Key in [#8, '0'..'9', '.']) then
  begin
    Key := #0;
  end
  else if (Key = '.') and (Pos(Key, edtInsQty.Text) > 0) then
  begin
    Key := #0;
  end;
end;

procedure TfrmJobInspect.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  if (Label2.Caption <> '0') then
  begin
    ShowMessage('You must complete this section!');
    Action := caNone;
  end;
end;
```

```
procedure TfrmJobInspect.FormShow(Sender: TObject);
var role : String;
begin
  btnAccept.Enabled    := False;
  btnScrap.Enabled     := False;

  temp := 'SELECT Role FROM [AdmUser] WHERE UserName = ' +
          QuotedStr(frmMenu.Label3.Caption);
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;

  role := qrySQL.FieldByName('Role').AsString;

  temp := 'SELECT * FROM [Authority] WHERE Role = ' +
          QuotedStr(role);
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;

  with qrySQL do
  begin
    if FieldByName('ValInputSerialNumber').AsBoolean = true then
      btnAccept.Enabled := True;
    if FieldByName('ValInputDataScrap').AsBoolean = true then
      btnScrap.Enabled := True;
  end;
  label2.Caption := '0';
  bersihkan;
end;

procedure TfrmJobInspect.bersihkan;
begin
  edtDtlJob.Text:='';
  edtDtlJobDesc.Text:='';
```

```
  edtDtlStockCode.Text:='';
  edtDtlStockDesc.Text:='';
  edtDtlWh.Text:='';
  edtDtlWhDesc.Text:='';
  edtQtyManuf.Text:='';
  edtQtyInsp.Text:='';
  edtQtyScrapped.Text:='';
  edtQtyAcc.Text:='';
  edtComplete.Text:='';
  edtInsUom.Text:='';
  edtInsQty.Text:='0';
  ckInsComplete.Checked:=false;
  edtInsQty.Enabled := true;
  cbInspect.Clear;
  cbInspect.Items.Clear;
  cbInspect.ClearSelection;

  temp := 'SELECT InspectCode FROM [Pro_Inspect] Where InspectCompleted
<> ' +
        QuotedStr('Y');
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;

  while not qrySQL.Eof do
  begin
    cbInspect.Items.Add(qrySQL.FieldByName('InspectCode').AsString);
    qrySQL.Next;
  end;
  cbInspect.ItemIndex := -1;
end;

end.
```

FormCreateSerialNumber.pas

```
unit FormCreateSerialNumber;

interface
```

```
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Buttons, Vcl.StdCtrls, Vcl.Mask,
  AdvDropDown, AdvCustomGridDropDown, AdvGridDropDown, Data.DB,
Data.Win.ADODB,
  Vcl.Grids, AdvObj, BaseGrid, AdvGrid, DBAdvGrid, StrUtils;

type
  TfrmCreateSerialNumber = class(TForm)
    GroupBox1: TGroupBox;
    Edit18: TEdit;
    Edit19: TEdit;
    edtAccUom: TEdit;
    edtAccQty: TEdit;
    Memo1: TMemo;
    memo: TMemo;
    btnPost: TSpeedButton;
    Edit1: TEdit;
    edtSerialNumber: TEdit;
    btnSubmit: TButton;
    dsAcc: TDataSource;
    qryAcc: TADOQuery;
    qrySQL: TADOQuery;
    qrySQL2: TADOQuery;
    qryAccStockCode: TWideStringField;
    qryAccSerialNumber: TWideStringField;
    qryAccWarehouse: TWideStringField;
    qryAccQty: TFMTBCDField;
    qryAccUOM: TWideStringField;
    qryAccReference: TWideStringField;
    qryAccNotes: TWideStringField;
    qryAccStatus: TWideStringField;
    btnDelete: TButton;
    DBAdvGrid2: TDBAdvGrid;
    qryAccReference2: TWideStringField;

    function CekSubmit : Boolean;
    procedure FormShow(Sender: TObject);
    procedure edtAccQtyKeyPress(Sender: TObject; var Key: Char);
    procedure bersihkan;
    procedure btnSubmitClick(Sender: TObject);
    procedure btnPostClick(Sender: TObject);
```

```
    procedure btnDeleteClick(Sender: TObject);
    procedure edtAccQtyExit(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmCreateSerialNumber: TfrmCreateSerialNumber;
  temp : String;
  totalInspect, qtyInspect : Double;

implementation
uses FormComponent, FormJobInspect, Numbering;

{$R *.dfm}

function TfrmCreateSerialNumber.CekSubmit : Boolean;
begin
  CekSubmit := True;
  qtyInspect := StrToFloat(edtAccQty.Text);

  if (edtAccQty.Text = '') or (edtAccQty.Text = '0') then
  begin
    MessageDlg('Quantity Accepted required!', mtError, [mbOK], 0);
    edtAccQty.SetFocus;
    CekSubmit := False;
  end
  else
  if qtyInspect > totalInspect then
  begin
    edtAccQty.Text := '0';
    MessageDlg('Max. '+ FloatToStr(totalInspect), mtError, [mbOK], 0);
    edtAccQty.SetFocus;
    CekSubmit := False;
  end;
end;

procedure TfrmCreateSerialNumber.btnDeleteClick(Sender: TObject);
begin
  if not qryAcc.Eof then
  begin
```

```
    frmJobInspect.Label2.Caption :=
FloatToStr(totalInspect+qryAcc.FieldByName('Qty').AsFloat);
    qryAcc.Delete;
    qryAcc.Close;
    qryAcc.Parameters[0].Value := frmJobInspect.cbInspect.Text;
    qryAcc.Parameters[1].Value := frmJobInspect.edtDtlJob.Text;
    qryAcc.Open;
  end;
end;

procedure TfrmCreateSerialNumber.btnPostClick(Sender: TObject);
var totalqty : Double;
begin
  if not qryAcc.Eof then
  begin
    temp := 'Select sum(Qty) as total from [InventorySerial] '+
          'where Reference = ' + QuotedStr(frmJobInspect.cbInspect.Text)+
          ' and Reference2 = '+ QuotedStr(frmJobInspect.edtDtlJob.Text)+
          ' and Status <> '+ QuotedStr('POST');
    with qrySQL do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
    end;
    totalqty := qrySQL.FieldByName('total').AsFloat;

    if not frmComponent.adoFauzi.InTransaction then
    begin
      frmComponent.adoFauzi.BeginTrans;
      try
        temp := 'Select * from [Pro_Inspect] where InspectCode = ' +
              QuotedStr(frmJobInspect.cbInspect.Text);
        with qrySQL do
        begin
          Close;
          SQL.Clear;
          SQL.Add(temp);
          Open;
```

```
        Edit;
FieldByName('QtyAccepted').AsFloat:=FieldByName('QtyAccepted').AsFloat+t
otalqty;
        Post;
      end;

      temp:= 'SELECT * FROM Pro_Job where Job = ' +
QuotedStr(frmJobInspect.edtDtlJob.Text);
      with qrySQL2 do
      begin
        Close;
        SQL.Clear;
        SQL.Add(temp);
        Open;
        Edit;
FieldByName('QtyAccepted').AsFloat:=FieldByName('QtyAccepted').AsFloat+t
otalqty;
        Post;
      end;

      temp:= 'SELECT * FROM InventorySerial where Reference = ' +
QuotedStr(frmJobInspect.cbInspect.Text)+
         ' and Reference2 = '+ QuotedStr(frmJobInspect.edtDtlJob.Text)+
         ' and Status = '+ QuotedStr('TEMP');
      with qrySQL2 do
      begin
        Close;
        SQL.Clear;
        SQL.Add(temp);
        Open;
        while not Eof do
        begin
          Edit;
          FieldByName('Status').AsString := 'POST';
          Post;
          Next;
        end;
      end;

      temp := 'SELECT * FROM [InventoryWh] Where StockCode = '+
            QuotedStr(frmJobInspect.edtDtlStockCode.Text)+
            ' and Warehouse = '+
            QuotedStr(frmJobInspect.edtDtlWh.Text);
```

```
        with qrySQL2 do
        begin
          Close;
          SQL.Clear;
          SQL.Add(temp);
          Open;
          Edit;
          FieldByName('QtyInInspection').AsFloat:=
FieldByName('QtyInInspection').AsFloat-totalqty;
          Post;
        end;
        MessageDlg('Data Posted!',mtConfirmation,[mbOK],0);
        frmComponent.adoFauzi.CommitTrans;
        PostMessage(Self.Handle,wm_close,0,0);
        bersihkan;
        Except
        on E : Exception do
        begin
          frmComponent.adoFauzi.RollbackTrans;
          MessageDlg(E.Message, mtError, [mbOK], 0);
        end;
      end;
    end;
  end
  else
  begin
    ShowMessage('There is an empty data!');
  end;
end;

procedure TfrmCreateSerialNumber.btnSubmitClick(Sender: TObject);
var qty : Double;
begin
  if CekSubmit then
  begin
    temp:= 'SELECT * FROM InventorySerial where SerialNumber = ' +
           QuotedStr(edtSerialNumber.Text);
    with qrySQL do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
```

```
    if not frmComponent.adoFauzi.InTransaction then
    begin
      frmComponent.adoFauzi.BeginTrans;
      try
        Qty := StrToFloat(StringReplace(edtAccQty.Text,',',',[rfReplaceAll,
rfIgnoreCase]));

        Insert;

FieldByName('StockCode').AsString:=frmJobInspect.edtDtlStockCode.Text;
        FieldByName('SerialNumber').AsString:=edtSerialNumber.Text;
        FieldByName('Warehouse').AsString:=frmJobInspect.edtDtlWh.Text;
        FieldByName('Qty').AsFloat:=qty;
        FieldByName('UOM').AsString:=edtAccUom.Text;
        FieldByName('Reference').AsString:=frmJobInspect.cbInspect.Text;
        FieldByName('Reference2').AsString:=frmJobInspect.edtDtlJob.Text;
        FieldByName('Notes').AsString:=memo.Text;
        FieldByName('Status').AsString:='TEMP';
        FieldByName('AddId').AsString := frmComponent.lblUserName.Caption;
        Post;

        frmJobInspect.Label2.Caption := FloatToStr(totalInspect-qty);

        qryAcc.Close;
        qryAcc.Parameters[0].Value := frmJobInspect.cbInspect.Text;
        qryAcc.Parameters[1].Value := frmJobInspect.edtDtlJob.Text;
        qryAcc.Open;

        frmComponent.adoFauzi.CommitTrans;
        bersihkan;

        Except
        on E : Exception do
        begin
          frmComponent.adoFauzi.RollbackTrans;
          MessageDlg(E.Message, mtError, [mbOK], 0);
        end;
      end;
    end;
  end;
 end;
end;
```

```
procedure TfrmCreateSerialNumber.edtAccQtyExit(Sender: TObject);
begin
  qtyInspect := StrToFloat(edtAccQty.Text);
  if qtyInspect > totalInspect then
  begin
    edtAccQty.Text := '0';
    edtAccQty.SetFocus;
    ShowMessage('Max. '+ FloatToStr(totalInspect));
  end;
end;

procedure TfrmCreateSerialNumber.edtAccQtyKeyPress(Sender: TObject;
  var Key: Char);
begin
  if not (Key in [#8, '0'..'9', '.']) then
  begin
    Key := #0;
  end
  else if (Key = '.') and (Pos(Key, edtAccQty.Text) > 0) then
  begin
    Key := #0;
  end;
end;

procedure TfrmCreateSerialNumber.FormShow(Sender: TObject);
begin
  bersihkan;
end;

procedure TfrmCreateSerialNumber.bersihkan;
var Panjang, NextSuffix : Integer;
    Prefix, temp_left : String;
begin
  edtAccUom.Text := frmJobInspect.edtInsUom.Text;
  edtAccQty.Text := '0';
  memo.Clear;
  totalInspect := StrToFloat(frmJobInspect.Label2.Caption);

  temp := 'select Top 1 SerialNumber From InventorySerial order by
SerialNumber DESC';
  with qrySQL do
  begin
    Close;
```

```
  SQL.Clear;
  SQL.Add(temp);
  Open;
 end;

 if qrySQL.Eof then
 begin
  NextSuffix := 1;
 end
 else
 begin
  qrySQL.Last;
  temp_left  := RightStr(qrySQL.FieldByName('SerialNumber').AsString, 6);
  NextSuffix := StrToInt(temp_left)+ 1;
 end;

 Prefix  := 'SN';
 Panjang := 8;
 edtSerialNumber.Text := UpdateCode(IntToStr(NextSuffix),Prefix,Panjang);

 qryAcc.Close;
 qryAcc.Parameters[0].Value := frmJobInspect.cbInspect.Text;
 qryAcc.Parameters[1].Value := frmJobInspect.edtDtlJob.Text;
 qryAcc.Open;
end;

end.
```

FormInputDataScrap.pas

```
unit FormInputDataScrap;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Buttons, Vcl.Mask, AdvDropDown,
  AdvCustomGridDropDown, AdvGridDropDown, Vcl.StdCtrls, Data.DB,
Data.Win.ADODB;

type
  TfrmInputDataScrap = class(TForm)
    GroupBox1: TGroupBox;
```

```
    Edit18: TEdit;
    Edit19: TEdit;
    edtScrUom: TEdit;
    edtScrpQty: TEdit;
    btnPost: TSpeedButton;
    Memo1: TMemo;
    memo: TMemo;
    qrySQL: TADOQuery;
    qrySQL2: TADOQuery;

    function CekSave : Boolean;
    procedure FormShow(Sender: TObject);
    procedure btnPostClick(Sender: TObject);
    procedure edtScrpQtyExit(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmInputDataScrap: TfrmInputDataScrap;
  temp : String;
  totalInspect, qtyInspect : Double;

implementation
uses FormJobInspect, FormComponent;

{$R *.dfm}

procedure TfrmInputDataScrap.btnPostClick(Sender: TObject);
var qty, QtyonHand, UnitCost, UnitCostBaru : Double;
begin
  if CekSave then
  begin
    temp := 'SELECT * FROM [Pro_Inspect] Where InspectCode = ' +
          QuotedStr(frmJobInspect.cbInspect.Text);
    with qrySQL do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
```

```
    if not frmComponent.adoFauzi.InTransaction then
    begin
      frmComponent.adoFauzi.BeginTrans;
      try
        Qty := StrToFloat(StringReplace(edtScrpQty.Text,',',',[rfReplaceAll,
rfIgnoreCase]));

        Edit;
FieldByName('QtyScrapped').AsFloat:=FieldByName('QtyScrapped').AsFloat+
qty;
        Post;

        temp:= 'SELECT * FROM Pro_Job where Job = ' +
QuotedStr(frmJobInspect.edtDtlJob.Text);
        with qrySQL2 do
        begin
          Close;
          SQL.Clear;
          SQL.Add(temp);
          Open;
          Edit;
FieldByName('QtyScrapped').AsFloat:=FieldByName('QtyScrapped').AsFloat+
Qty;
          Post;
        end;

        temp := 'SELECT * FROM [InventoryWh] Where StockCode = '+
              QuotedStr(frmJobInspect.edtDtlStockCode.Text)+
              ' and Warehouse = '+
              QuotedStr(frmJobInspect.edtDtlWh.Text);
        with qrySQL2 do
        begin
          Close;
          SQL.Clear;
          SQL.Add(temp);
          Open;
          UnitCost := FieldByName('UnitCost').AsFloat;
          QtyonHand:= FieldByName('QtyOnHand').AsFloat;
          UnitCostBaru := (QtyonHand*UnitCost)/(QtyonHand-qty);

          Edit;
          FieldByName('QtyInInspection').AsFloat:=
FieldByName('QtyInInspection').AsFloat-qty;
```

```
          FieldByName('QtyOnHand').AsFloat:=
FieldByName('QtyOnHand').AsFloat-qty;
          FieldByName('UnitCost').AsFloat:= UnitCostBaru;
          Post;
        end;

        temp := 'SELECT * FROM [InventoryWh] Where StockCode = '+
              QuotedStr(frmJobInspect.edtDtlStockCode.Text)+
              ' and Warehouse = '+
              QuotedStr('WH004');
        with qrySQL2 do
        begin
          Close;
          SQL.Clear;
          SQL.Add(temp);
          Open;
          Edit;
          FieldByName('QtyOnHand').AsFloat:=
FieldByName('QtyOnHand').AsFloat+qty;
          Post;
        end;

        temp := 'SELECT * FROM [InventoryScrap] Where Reference = ' +
              QuotedStr(frmJobInspect.cbInspect.Text);
        with qrySQL2 do
        begin
          Close;
          SQL.Clear;
          SQL.Add(temp);
          Open;

          Insert;
     FieldByName('StockCode').AsString:=frmJobInspect.edtDtlStockCode.Text;
          FieldByName('Qty').AsFloat :=qty;
          FieldByName('Reference').AsString := frmJobInspect.cbInspect.Text;
          FieldByName('Reference2').AsString := frmJobInspect.edtDtlJob.Text;
          FieldByName('Notes').AsString := memo.Text;
          FieldByName('AddId').AsString :=
frmComponent.lblUserName.Caption;
          Post;
        end;

        frmJobInspect.Label2.Caption := FloatToStr(totalInspect-qty);
```

```
        MessageDlg('Post success!',mtConfirmation,[mbOK],0);
        frmComponent.adoFauzi.CommitTrans;
        PostMessage(Self.Handle,wm_close,0,0);

        Except
        on E : Exception do
        begin
          frmComponent.adoFauzi.RollbackTrans;
          MessageDlg(E.Message, mtError, [mbOK], 0);
        end;
      end;
    end;
  end;
 end;
end;

function TfrmInputDataScrap.CekSave : Boolean;
begin
  CekSave := True;
  qtyInspect := StrToFloat(edtScrpQty.Text);

  if (edtScrpQty.Text = '') or (edtScrpQty.Text = '0') then
  begin
    MessageDlg('Quantity Scrapped required!', mtError, [mbOK], 0);
    edtScrpQty.SetFocus;
    CekSave := False;
  end
  else
  if qtyInspect > totalInspect then
  begin
    edtScrpQty.Text := '0';
    MessageDlg('Max. '+ FloatToStr(totalInspect), mtError, [mbOK], 0);
    edtScrpQty.SetFocus;
    CekSave := False;
  end;
end;

procedure TfrmInputDataScrap.edtScrpQtyExit(Sender: TObject);
begin
  qtyInspect := StrToFloat(edtScrpQty.Text);
  if qtyInspect > totalInspect then
  begin
    edtScrpQty.Text := '0';
```

```
    edtScrpQty.SetFocus;
    ShowMessage('Max. '+ FloatToStr(totalInspect));
  end;
end;

procedure TfrmInputDataScrap.FormShow(Sender: TObject);
begin
  edtScrUom.Text := frmJobInspect.edtInsUom.Text;
  edtScrpQty.Text := '0';
  totalInspect := StrToFloat(frmJobInspect.Label2.Caption);
  memo.Clear;
end;

end.
```

FormViewInspectJob.pas

```
unit FormViewInspectJob;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Buttons, Vcl.Grids,
  AdvObj, BaseGrid, AdvGrid, DBAdvGrid, Data.DB, Data.Win.ADODB;

type
  TfrmViewInspectJob = class(TForm)
    GroupBox1: TGroupBox;
    DBAdvGrid1: TDBAdvGrid;
    qryInspect: TADOQuery;
    dsInspect: TDataSource;
    GroupBox2: TGroupBox;
    DBAdvGrid2: TDBAdvGrid;
    GroupBox3: TGroupBox;
    DBAdvGrid3: TDBAdvGrid;
    qryInspectInspectCode: TWideStringField;
    qryInspectJob: TWideStringField;
    qryInspectJobDescription: TWideStringField;
    qryInspectStockCode: TWideStringField;
    qryInspectWarehouse: TWideStringField;
    qryInspectReceiptDate: TDateTimeField;
    qryInspectQtyInspected: TFMTBCDField;
```

```
qryInspectQtyAccepted: TFMTBCDField;
qryInspectQtyScrapped: TFMTBCDField;
qryInspectInspectCompleted: TStringField;
qryInspectUnitCost: TFMTBCDField;
qryInspectOriginalValue: TBCDField;
qryInspectCostUom: TWideStringField;
qryInspectUom: TWideStringField;
qryInspectType: TStringField;
qryInspectTimeStamp: TBytesField;
qrySerial: TADOQuery;
dsSerial: TDataSource;
qrySerialStockCode: TWideStringField;
qrySerialSerialNumber: TWideStringField;
qrySerialWarehouse: TWideStringField;
qrySerialQty: TFMTBCDField;
qrySerialUOM: TWideStringField;
qrySerialReference: TWideStringField;
qrySerialNotes: TWideStringField;
qrySerialStatus: TWideStringField;
qrySerialTimeStamp: TBytesField;
qryScrap: TADOQuery;
dsScrap: TDataSource;
qryScrapStockCode: TWideStringField;
qryScrapQty: TFMTBCDField;
qryScrapUOM: TWideStringField;
qryScrapReference: TWideStringField;
qryScrapNotes: TWideStringField;
qryScrapReference2: TWideStringField;
qrySerialReference2: TWideStringField;
qryInspectWHDesc: TWideStringField;
qryInspectStockDesc: TWideStringField;
GroupBox4: TGroupBox;
btnInspect: TSpeedButton;
qryInspectnama1: TWideStringField;
qryInspectnama2: TWideStringField;
qrySerialName: TWideStringField;
qrySerialWHDesc: TWideStringField;
qrySerialStockDesc: TWideStringField;
qryScrapStockDesc: TWideStringField;
qryScrapName: TWideStringField;
qrySQL: TADOQuery;
qryInspectQtyReceipt: TFMTBCDField;
procedure btnInspectClick(Sender: TObject);
```

```
    procedure FormShow(Sender: TObject);
    procedure DBAdvGrid1ClickCell(Sender: TObject; ARow, ACol: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmViewInspectJob: TfrmViewInspectJob;

implementation
uses FormComponent, FormJobInspect, FormMenu;

{$R *.dfm}

procedure TfrmViewInspectJob.btnInspectClick(Sender: TObject);
begin
  frmJobInspect.ShowModal;
  qryInspect.Close;
  qryInspect.Open;
end;

procedure TfrmViewInspectJob.DBAdvGrid1ClickCell(Sender: TObject;
ARow,
  ACol: Integer);
begin
  with qrySerial do
  begin
    Close;
    Parameters[0].Value := qryInspect.FieldByName('InspectCode').AsString;
    Open;
  end;

  with qryScrap do
  begin
    Close;
    Parameters[0].Value := qryInspect.FieldByName('InspectCode').AsString;
    Open;
  end;
end;
```

```
procedure TfrmViewInspectJob.FormShow(Sender: TObject);
var role : String;
begin
  btnInspect.Enabled    := False;

  temp := 'SELECT Role FROM [AdmUser] WHERE UserName = ' +
          QuotedStr(frmMenu.Label3.Caption);
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;

  role := qrySQL.FieldByName('Role').AsString;

  temp := 'SELECT * FROM [Authority] WHERE Role = ' +
          QuotedStr(role);
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;

  with qrySQL do
  begin
    if FieldByName('ValJobInspection').AsBoolean = true then
      btnInspect.Enabled := True;
  end;

  qryInspect.Close;
  qryInspect.Open;

  qrySerial.Close;
  qryScrap.Close;
end;

end.
```