# LAMPIRAN

**FromData.pas**

```
unit FormUtama;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.StdCtrls,
  Vcl.Imaging.pngimage;

type
  TFrmUtama = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    imgHome: TImage;
    imgData: TImage;
    imgUsers: TImage;
    imgLogout: TImage;
    Home: TLabel;
    Data: TLabel;
    Users: TLabel;
    Label4: TLabel;
    Label1: TLabel;
    procedure FormShow(Sender: TObject);
    procedure imgHomeClick(Sender: TObject);
    procedure imgDataClick(Sender: TObject);
    procedure imgUsersClick(Sender: TObject);
    procedure imgLogoutClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  FrmUtama: TFrmUtama;
implementation
{$R *.dfm}
uses FormLogin, FormHome, FormData, FormUsers;
procedure TFrmUtama.FormShow(Sender: TObject);
var LoginStatus: boolean;
begin
  frmLogin := TFrmLogin.Create(self);
  frmLogin.ShowModal;
  LoginStatus := frmLogin.LoginStatus;
  if LoginStatus = False then
  begin
    Application.Terminate;
  end;
end;
```

```
procedure TFrmUtama.imgHomeClick(Sender: TObject);
begin
  TFrmHome.Create(NIL);
end;
procedure TFrmUtama.imgDataClick(Sender: TObject);
begin
  TFrmData.Create(NIL);
end;
procedure TFrmUtama.imgUsersClick(Sender: TObject);
begin
  TFrmUsers.Create(NIL);
end;

procedure TFrmUtama.imgLogoutClick(Sender: TObject);
begin
  if MessageDlg('Yakin ingin logout?', mtConfirmation, [mbYes, mbNo], 0) = mrYes then
  Application.Terminate;
end;
end.
```

**FormHome.pas**

```
unit FormUtama;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.StdCtrls,
  Vcl.Imaging.pngimage;
type
  TFrmUtama = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    imgHome: TImage;
    imgData: TImage;
    imgUsers: TImage;
    imgLogout: TImage;
    Home: TLabel;
    Data: TLabel;
    Users: TLabel;
    Label4: TLabel;
    Label1: TLabel;
    procedure FormShow(Sender: TObject);
    procedure imgHomeClick(Sender: TObject);
    procedure imgDataClick(Sender: TObject);
    procedure imgUsersClick(Sender: TObject);
    procedure imgLogoutClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
  end;

var

  FrmUtama: TFrmUtama;

implementation

{$R *.dfm}

uses FormLogin, FormHome,
FormData, FormUsers;

procedure
TFrmUtama.FormShow(Sender:
TObject);

var LoginStatus: boolean;

begin

  frmLogin := TFrmLogin.Create(self);

  frmLogin.ShowModal;

  LoginStatus := frmLogin.LoginStatus;

  if LoginStatus = False then

  begin

    Application.Terminate;

  end;

end;

procedure
TFrmUtama.imgHomeClick(Sender:
TObject);

begin

  TFrmHome.Create(NIL);

end;

procedure
TFrmUtama.imgDataClick(Sender:
TObject);

begin

  TFrmData.Create(NIL);

end;

procedure
TFrmUtama.imgUsersClick(Sender:
TObject);

begin

  TFrmUsers.Create(NIL);

end;

procedure
TFrmUtama.imgLogoutClick(Sender:
TObject);

begin

  if MessageDlg('Yakin ingin logout?',
mtConfirmation, [mbYes, mbNo], 0) =
mrYes then

  Application.Terminate;

end;

end.
```

**FormLaporan.pas**

```
unit FormLaporan;

interface

uses

  Winapi.Windows,

  Winapi.Messages,

  System.SysUtils,

  System.Variants,

  System.Classes,

  Vcl.Graphics,

  Vcl.Controls,

  Vcl.Forms,

  Vcl.Dialogs,

  FireDAC.Stan.Intf,

  FireDAC.Stan.Option,

  FireDAC.Stan.Param,
```

```
  FireDAC.Stan.Error,
  FireDAC.DatS,
  FireDAC.Phys.Intf,
  FireDAC.DApt.Intf,
  FireDAC.Stan.Async,
  FireDAC.DApt,
  FireDAC.UI.Intf,
  FireDAC.Stan.Def,
  FireDAC.Stan.Pool,
  FireDAC.Phys,
  FireDAC.Phys.SQLite,
  FireDAC.Phys.SQLiteDef,
  FireDAC.Stan.ExprFuncs,
  Data.DB,
  FireDAC.Comp.Client,
  FireDAC.Comp.DataSet,
  Vcl.StdCtrls,
  frxClass,
  frxDBSet,
  DateUtils;
type
  TfrmLaporan = class(TForm)
    ComboBoxBulan: TComboBox;
    Label1: TLabel;
    Label2: TLabel;
    ComboBoxTahun: TComboBox;
    ButtonLaporanTahunan: TButton;
    FDConnectionPeminjaman:
TFDConnection;
    ButtonLaporanBulanan: TButton;
    frxDBDataset1: TfrxDBDataset;
    frxReport1: TfrxReport;
    FDQueryPeminjaman: TFDQuery;
    procedure
ButtonLaporanBulananClick(Sender:
TObject);
    procedure
ButtonLaporanTahunanClick(Sender:
TObject);
    procedure FormCreate(Sender:
TObject);
  private
    procedure LogAndAlertIfEmpty(const
Context: string);
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmLaporan: TfrmLaporan;
implementation
{$R *.dfm}
procedure
TfrmLaporan.FormCreate(Sender:
TObject);
var
  i, TahunSekarang: Integer;
begin
  // Isi bulan 1..12
  ComboBoxBulan.Items.Clear;
  for i := 1 to 12 do
```

```
ComboBoxBulan.Items.Add(IntToStr(i));
  // Isi tahun (Now-2)..Now
  ComboBoxTahun.Items.Clear;
  TahunSekarang := YearOf(Date);
  for i := TahunSekarang - 2 to TahunSekarang do
ComboBoxTahun.Items.Add(IntToStr(i));
  // Default pilih sekarang
  ComboBoxBulan.ItemIndex := MonthOf(Date) - 1;
  ComboBoxTahun.ItemIndex := ComboBoxTahun.Items.Count - 1;
end;
procedure TfrmLaporan.ButtonLaporanBulananClick(Sender: TObject);
var
  Bulan, Tahun: Integer;
begin
  if (ComboBoxBulan.ItemIndex < 0) or (ComboBoxTahun.ItemIndex < 0) then
  begin
    ShowMessage('Pilih bulan dan tahun terlebih dahulu.');
    Exit;
  end;
  Bulan := ComboBoxBulan.ItemIndex + 1;
  Tahun := StrToInt(ComboBoxTahun.Text);
  FDQueryPeminjaman.Close;
  FDQueryPeminjaman.SQL.Text :=
    'SELECT p.id_peminjaman, p.id_mahasiswa, m.nim, m.nama AS nama_mahasiswa, '+
    'b.id_buku, b.judul, b.penulis, p.tanggal_pinjam, p.tanggal_kembali, p.status '+
    'FROM peminjaman p '+
    'JOIN mahasiswa m ON p.id_mahasiswa = m.id_mahasiswa '+
    'JOIN buku b ON p.id_buku = b.id_buku '+
    // substr(tanggal, posisi_mulai, panjang)
    'WHERE CAST(substr(p.tanggal_pinjam, 4, 2) AS INTEGER) = :bulan '+
     'AND CAST(substr(p.tanggal_pinjam, 7, 4) AS INTEGER) = :tahun';
FDQueryPeminjaman.ParamByName('bulan').AsInteger := Bulan;
FDQueryPeminjaman.ParamByName('tahun').AsInteger := Tahun;
  FDQueryPeminjaman.Open;
  LogAndAlertIfEmpty(Format('Laporan Bulanan %d-%2.2d', [Tahun, Bulan]));
  frxReport1.ShowReport;
end;
procedure TfrmLaporan.ButtonLaporanTahunanClick(Sender: TObject);
var
```

```
  Tahun: Integer;
begin
  if ComboBoxTahun.ItemIndex < 0 then
  begin
    ShowMessage('Pilih tahun terlebih dahulu.');
    Exit;
  end;
  Tahun := StrToInt(ComboBoxTahun.Text);
  FDQueryPeminjaman.Close;
  FDQueryPeminjaman.SQL.Text :=
    'SELECT p.id_peminjaman, p.id_mahasiswa, m.nim, m.nama AS nama_mahasiswa, '+
    'b.id_buku, b.judul, b.penulis, p.tanggal_pinjam, p.tanggal_kembali, p.status '+
    'FROM peminjaman p '+
    'JOIN mahasiswa m ON p.id_mahasiswa = m.id_mahasiswa '+
    'JOIN buku b ON p.id_buku = b.id_buku '+
    'WHERE CAST(substr(p.tanggal_pinjam, 7, 4) AS INTEGER) = :tahun';

FDQueryPeminjaman.ParamByName('tahun').AsInteger := Tahun;
  FDQueryPeminjaman.Open;
  LogAndAlertIfEmpty(Format('Laporan Tahunan %d', [Tahun]));
  frxReport1.ShowReport;
end;

procedure TfrmLaporan.LogAndAlertIfEmpty(const Context: string);
var
  cnt: Integer;
begin
  cnt := FDQueryPeminjaman.RecordCount;
  OutputDebugString(PChar(Context + ' – RecordCount=' + IntToStr(cnt)));
  if cnt = 0 then
  begin
    ShowMessage(Context + ': tidak ada data.');
    Abort;  // hentikan ShowReport
  end;
end;
end.
```

**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.StdCtrls, FireDAC.Stan.Intf,
  FireDAC.Stan.Option, FireDAC.Stan.Param, FireDAC.Stan.Error, FireDAC.DatS,
  FireDAC.Phys.Intf, FireDAC.DApt.Intf, FireDAC.Stan.Async, FireDAC.DApt,
```

```
  FireDAC.UI.Intf, FireDAC.Stan.Def,
FireDAC.Stan.Pool, FireDAC.Phys,
  FireDAC.Phys.SQLite,
FireDAC.Phys.SQLiteDef,
FireDAC.Stan.ExprFuncs, Data.DB,
  FireDAC.Comp.Client,
FireDAC.Comp.DataSet;
type
  TFrmLogin = class(TForm)
    btnLogin: TButton;
    EditUsername: TEdit;
    EditPassword: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    FDQuery1: TFDQuery;
    FDConnection1: TFDConnection;
    procedure btnLoginClick(Sender:
TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    LoginStatus: boolean;
  end;
var
  FrmLogin: TFrmLogin;
implementation
{$R *.dfm}
procedure
TFrmLogin.btnLoginClick(Sender:
TObject);
begin
  if (EditUsername.Text = '') or
(EditPassword.Text = '') then
  begin
    ShowMessage('Username dan
Password harus diisi!');
    Exit;
  end;
  FDQuery1.SQL.Text := 'SELECT *
FROM users WHERE username =
:username AND password = :password';

FDQuery1.ParamByName('username').
AsString := EditUsername.Text;

FDQuery1.ParamByName('password').A
sString := EditPassword.Text;
  FDQuery1.Open;
  if not FDQuery1.IsEmpty then
  begin
    LoginStatus := True;
    Close;
  end
  else
  begin
    ShowMessage('Username atau
Password salah!');
    EditPassword.Clear;
    EditPassword.SetFocus;
    LoginStatus := False;
  end;
end;
end.
```

**FormScannerBuku.pas**

```
unit FormScannerBuku;

interface

uses

  Winapi.Windows, Winapi.Messages,
System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,

  Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
Vcl.StdCtrls, Vcl.ExtCtrls, VFrames,
VSample,

  ZXing.ReadResult,
ZXing.BarCodeFormat,
ZXing.ScanManager, Vcl.ComCtrls,

  FireDAC.Stan.Intf,
FireDAC.Stan.Option,
FireDAC.Stan.Param,

  FireDAC.Stan.Error, FireDAC.DatS,
FireDAC.Phys.Intf, FireDAC.DApt.Intf,

  FireDAC.Stan.Async, FireDAC.DApt,
Data.DB, FireDAC.Comp.DataSet,

  FireDAC.Comp.Client,
FireDAC.UI.Intf, FireDAC.Stan.Def,
FireDAC.Stan.Pool,

  FireDAC.Phys, FireDAC.Phys.SQLite,
FireDAC.Phys.SQLiteDef,

  FireDAC.Stan.ExprFuncs;

type

  TFrmScannerBuku = class(TForm)

    Image1: TImage;

    FDQuery1: TFDQuery;

    Label1: TLabel;

    LabelScanned: TLabel;

    FDConnection1: TFDConnection;

    procedure FormCreate(Sender:
TObject);

    procedure FormDestroy(Sender:
TObject);

    procedure FormDeactivate(Sender:
TObject);

    procedure FormActivate(Sender:
TObject);

  private

    fVideoImage: TVideoImage;

    procedure
NewVideoFrameEvent(Sender: TObject;
Width, Height: Integer; DataPtr:
Pointer);

  public

    idBukuScanned: String;

    judulBukuScanned: String;

    penulisBukuScanned: String;

  end;

var

  FrmScannerBuku: TFrmScannerBuku;

implementation

{$R *.dfm}

procedure
TFrmScannerBuku.FormDeactivate(Sen
der: TObject);

begin

  fVideoImage.VideoStop;

end;

procedure
TFrmScannerBuku.FormActivate(Sende
r: TObject);

var

  listcams: TStringList;

begin
```

```
  listcams := TStringList.Create;
  try

fVideoImage.GetListOfDevices(listcams
);
    if listcams.Count > 0 then
    begin

fVideoImage.VideoStart(listcams[0]);
    end;
  finally
    listcams.Free;
  end;
end;
procedure
TFrmScannerBuku.FormCreate(Sender:
TObject);
begin
  fVideoImage := TVideoImage.Create();
  fVideoImage.OnNewVideoFrame :=
NewVideoFrameEvent;
  Self.OnDeactivate := FormDeactivate;
  Self.OnActivate := FormActivate;
end;
procedure
TFrmScannerBuku.FormDestroy(Sender
: TObject);
begin
  fVideoImage.Free;
end;
procedure
TFrmScannerBuku.NewVideoFrameEve
nt(Sender: TObject; Width, Height:
Integer; DataPtr: Pointer);
var
  bitmap: TBitmap;
  ReadResult: TReadResult;
  ScanManager: TScanManager;
  Barcode: String;
begin
  bitmap := TBitmap.Create;
  try
    bitmap.PixelFormat := pfDevice;
    fVideoImage.GetBitmap(bitmap);
    Image1.Picture.Assign(bitmap);
    // scan code
    try
      ScanManager :=
TScanManager.Create(TBarcodeFormat.
CODE_128, nil);
      ReadResult :=
ScanManager.Scan(bitmap);
      if Assigned(ReadResult) then
      begin
        barcode := ReadResult.text;
        LabelScanned.Caption := barcode;
        // query to get id_buku from table
buku where barcode :barcode
        FDQuery1.SQL.Text := 'SELECT *
FROM buku WHERE barcode =
:barcode';

FDQuery1.ParamByName('barcode').As
String := barcode;
        FDQuery1.Open;
        try
          if not FDQuery1.IsEmpty then
```

```
        begin
          idBukuScanned :=
FDQuery1.FieldByName('id_buku').AsS
tring;
          judulBukuScanned :=
FDQuery1.FieldByName('judul').AsStri
ng;
          penulisBukuScanned :=
FDQuery1.FieldByName('penulis').AsSt
ring;
          Close;
        end
        else
        begin
          idBukuScanned := ''; // or handle
the case where no matching barcode is
found
        end;
      finally
        FDQuery1.Close;
      end;
    end;
  finally
    FreeAndNil(ScanManager);
    FreeAndNil(ReadResult);
  end;
 finally
  bitmap.Free;
 end;
 Application.ProcessMessages;
end;
end.
```

**FormScannerMhs.pas**

```
unit FormScannerMhs;

interface

uses
  Winapi.Windows, Winapi.Messages,
System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
Vcl.StdCtrls, Vcl.ExtCtrls, VFrames,
VSample,
  ZXing.ReadResult,
ZXing.BarCodeFormat,
ZXing.ScanManager, Vcl.ComCtrls,
  FireDAC.Stan.Intf,
FireDAC.Stan.Option,
FireDAC.Stan.Param,
  FireDAC.Stan.Error, FireDAC.DatS,
FireDAC.Phys.Intf, FireDAC.DApt.Intf,
  FireDAC.Stan.Async, FireDAC.DApt,
Data.DB, FireDAC.Comp.DataSet,
  FireDAC.Comp.Client,
FireDAC.UI.Intf, FireDAC.Stan.Def,
FireDAC.Stan.Pool,
  FireDAC.Phys, FireDAC.Phys.SQLite,
FireDAC.Phys.SQLiteDef,
  FireDAC.Stan.ExprFuncs;
type
  TFrmScannerMhs = class(TForm)
    Image1: TImage;
    FDQuery1: TFDQuery;
    Label1: TLabel;
    LabelScanned: TLabel;
    FDConnection1: TFDConnection;
    procedure FormCreate(Sender:
TObject);
```

```
    procedure FormDestroy(Sender:
TObject);
    procedure FormDeactivate(Sender:
TObject);
    procedure FormActivate(Sender:
TObject);
  private
    fVideoImage: TVideoImage;
    procedure
NewVideoFrameEvent(Sender: TObject;
Width, Height: Integer; DataPtr:
Pointer);
  public
    idMhsScanned: String;
    nimMhsScanned: String;
    namaMhsScanned: String;
  end;
var
  FrmScannerMhs: TFrmScannerMhs;
implementation
{$R *.dfm}
procedure
TFrmScannerMhs.FormDeactivate(Send
er: TObject);
begin
  fVideoImage.VideoStop;
end;
procedure
TFrmScannerMhs.FormActivate(Sender
: TObject);
var
  listcams: TStringList;
begin
  listcams := TStringList.Create;
  try

fVideoImage.GetListOfDevices(listcams
);
    if listcams.Count > 0 then
    begin

fVideoImage.VideoStart(listcams[0]);
    end;
  finally
    listcams.Free;
  end;
end;
procedure
TFrmScannerMhs.FormCreate(Sender:
TObject);
begin
  fVideoImage := TVideoImage.Create();
  fVideoImage.OnNewVideoFrame :=
NewVideoFrameEvent;
  Self.OnDeactivate := FormDeactivate;
  Self.OnActivate := FormActivate;
end;
procedure
TFrmScannerMhs.FormDestroy(Sender:
TObject);
begin
  fVideoImage.Free;
end;
procedure
TFrmScannerMhs.NewVideoFrameEven
t(Sender: TObject; Width, Height:
Integer; DataPtr: Pointer);
```

```
var
  bitmap: TBitmap;
  ReadResult: TReadResult;
  ScanManager: TScanManager;
  Barcode: String;
begin
  bitmap := TBitmap.Create;
  try
    bitmap.PixelFormat := pfDevice;
    fVideoImage.GetBitmap(bitmap);
    Image1.Picture.Assign(bitmap);
    // scan code
    try
      ScanManager := TScanManager.Create(TBarcodeFormat.CODE_128, nil);
      ReadResult := ScanManager.Scan(bitmap);
      if Assigned(ReadResult) then
      begin
        barcode := ReadResult.text;
        LabelScanned.Caption := barcode;
        // query to get id_buku from table buku where barcode :barcode
        FDQuery1.SQL.Text := 'SELECT * FROM mahasiswa WHERE nim = :barcode';

FDQuery1.ParamByName('barcode').AsString := barcode;
        FDQuery1.Open;
        try
          if not FDQuery1.IsEmpty then
          begin
            idMhsScanned := FDQuery1.FieldByName('id_mahasiswa').AsString;
            nimMhsScanned := FDQuery1.FieldByName('nim').AsString;
            namaMhsScanned := FDQuery1.FieldByName('nama').AsString;
            Close;
          end
          else
          begin
            idMhsScanned := ''; // or handle the case where no matching barcode is found
          end;
        finally
          FDQuery1.Close;
        end;
      end;
    finally
      FreeAndNil(ScanManager);
      FreeAndNil(ReadResult);
    end;
  finally
    bitmap.Free;
  end;
  Application.ProcessMessages;
end;
end.
```

**FormScannerPengembalian.pas**

```
unit FormScannerPengembalian;


interface

uses

  Winapi.Windows, Winapi.Messages,
System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,

  Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
Vcl.StdCtrls, Vcl.ExtCtrls, VFrames,
VSample,

  ZXing.ReadResult,
ZXing.BarCodeFormat,
ZXing.ScanManager, Vcl.ComCtrls,

  FireDAC.Stan.Intf,
FireDAC.Stan.Option,
FireDAC.Stan.Param,

  FireDAC.Stan.Error, FireDAC.DatS,
FireDAC.Phys.Intf, FireDAC.DApt.Intf,

  FireDAC.Stan.Async, FireDAC.DApt,
Data.DB, FireDAC.Comp.DataSet,

  FireDAC.Comp.Client,
FireDAC.UI.Intf, FireDAC.Stan.Def,
FireDAC.Stan.Pool,

  FireDAC.Phys, FireDAC.Phys.SQLite,
FireDAC.Phys.SQLiteDef,

  FireDAC.Stan.ExprFuncs;

type

  TFrmScannerPengembalian =
class(TForm)

    Image1: TImage;

    FDQuery1: TFDQuery;

    Label1: TLabel;

    LabelScanned: TLabel;

    FDConnection1: TFDConnection;

    procedure FormCreate(Sender:
TObject);

    procedure FormDestroy(Sender:
TObject);

    procedure FormDeactivate(Sender:
TObject);

    procedure FormActivate(Sender:
TObject);

  private

    fVideoImage: TVideoImage;

    procedure
NewVideoFrameEvent(Sender: TObject;
Width, Height: Integer; DataPtr:
Pointer);

  public

    idPeminjamanScanned: String;

    idMhsScanned: String;

    idBukuScanned: String;

  end;

var

  FrmScannerPengembalian:
TFrmScannerPengembalian;

implementation

{$R *.dfm}

procedure
TFrmScannerPengembalian.FormDeacti
vate(Sender: TObject);

begin

  fVideoImage.VideoStop;

end;

procedure
TFrmScannerPengembalian.FormActiva
te(Sender: TObject);

var
```

```
  listcams: TStringList;
begin
  listcams := TStringList.Create;
  try

fVideoImage.GetListOfDevices(listcams
);
    if listcams.Count > 0 then
    begin

fVideoImage.VideoStart(listcams[0]);
    end;
  finally
    listcams.Free;
  end;
end;
procedure
TFrmScannerPengembalian.FormCreate
(Sender: TObject);
begin
  fVideoImage := TVideoImage.Create();
  fVideoImage.OnNewVideoFrame :=
NewVideoFrameEvent;
  Self.OnDeactivate := FormDeactivate;
  Self.OnActivate := FormActivate;
end;
procedure
TFrmScannerPengembalian.FormDestro
y(Sender: TObject);
begin
  fVideoImage.Free;
end;
procedure
TFrmScannerPengembalian.NewVideoF
rameEvent(Sender: TObject; Width,
Height: Integer; DataPtr: Pointer);
var
  bitmap: TBitmap;
  ReadResult: TReadResult;
  ScanManager: TScanManager;
  Barcode: String;
begin
  bitmap := TBitmap.Create;
  try
    bitmap.PixelFormat := pfDevice;
    fVideoImage.GetBitmap(bitmap);
    Image1.Picture.Assign(bitmap);
    // scan code
    try
      ScanManager :=
TScanManager.Create(TBarcodeFormat.
CODE_128, nil);
      ReadResult :=
ScanManager.Scan(bitmap);
      if Assigned(ReadResult) then
      begin
        barcode := ReadResult.text;
        LabelScanned.Caption := barcode;
        // query to get id_buku from table
buku where barcode :barcode
        FDQuery1.SQL.Text := 'SELECT *
FROM peminjaman p JOIN mahasiswa
m ON p.id_mahasiswa =
m.id_mahasiswa WHERE nim =
:barcode AND status = :status';
```

```
FDQuery1.ParamByName('barcode').AsString := barcode;

FDQuery1.ParamByName('status').AsString := 'Dipinjam';
      FDQuery1.Open;
      try
        if not FDQuery1.IsEmpty then
        begin
          idPeminjamanScanned := FDQuery1.FieldByName('id_peminjaman').AsString;
          idMhsScanned := FDQuery1.FieldByName('id_mahasiswa').AsString;
          idBukuScanned := FDQuery1.FieldByName('id_buku').AsString;
          Close;
        end
        else
        begin
          idPeminjamanScanned := '';
          idMhsScanned := '';
          idBukuScanned := '';
          Close;
        end;
      finally
        FDQuery1.Close;
      end;
    end;
  finally
    FreeAndNil(ScanManager);
    FreeAndNil(ReadResult);
  end;
 finally
  bitmap.Free;
 end;
 Application.ProcessMessages;
end;
end.
```

**FormUsers.pas**

```
unit FormUsers;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Data.DB,
  FireDAC.Stan.Intf, FireDAC.Stan.Option, FireDAC.Stan.Param,
  FireDAC.Stan.Error, FireDAC.DatS, FireDAC.Phys.Intf, FireDAC.DApt.Intf,
  FireDAC.Stan.Async, FireDAC.DApt, FireDAC.UI.Intf, FireDAC.Stan.Def,
  FireDAC.Stan.Pool, FireDAC.Phys, FireDAC.Phys.SQLite, FireDAC.Phys.SQLiteDef,
  FireDAC.Stan.ExprFuncs, FireDAC.Comp.Client, FireDAC.Comp.DataSet, Vcl.Grids,
  Vcl.DBGrids, Vcl.ExtCtrls, FireDAC.VCLUI.Wait, FireDAC.Comp.UI;
```

```
type
  TFrmUsers = class(TForm)
    EditFullname: TEdit;
    EditUsername: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    EditPassword: TEdit;
    BtnSimpan: TButton;
    BtnEdit: TButton;
    BtnHapus: TButton;
    BtnClear: TButton;
    Panel1: TPanel;
    DBGrid1: TDBGrid;
    FDQuery1: TFDQuery;
    FDConnection1: TFDConnection;
    DataSource1: TDataSource;
    FDGUIxWaitCursor1:
TFDGUIxWaitCursor;
    procedure BtnSimpanClick(Sender:
TObject);
    procedure BtnEditClick(Sender:
TObject);
    procedure BtnHapusClick(Sender:
TObject);
    procedure BtnClearClick(Sender:
TObject);
    procedure
DBGrid1CellClick(Column: TColumn);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  FrmUsers: TFrmUsers;
implementation
{$R *.dfm}
procedure
TFrmUsers.BtnSimpanClick(Sender:
TObject);
begin
  if (EditFullname.Text = '') or
(EditUsername.Text = '') or
(EditPassword.Text = '') then
  begin
    ShowMessage('Semua kolom harus
diisi!');
    Exit;
  end;
  FDQuery1.SQL.Text := 'INSERT
INTO users (fullname, username,
password) VALUES (:fullname,
:username, :password)';

FDQuery1.ParamByName('fullname').A
sString := EditFullname.Text;

FDQuery1.ParamByName('username').
AsString := EditUsername.Text;

FDQuery1.ParamByName('password').A
sString := EditPassword.Text;
  FDQuery1.ExecSQL;
  FDQuery1.Close;
  FDQuery1.SQL.Text := 'SELECT *
FROM users';
```

```
  FDQuery1.Open;
  ShowMessage('Data berhasil disimpan!');
end;
procedure TFrmUsers.DBGrid1CellClick(Column: TColumn);
begin
  if not FDQuery1.IsEmpty then
  begin
    EditFullname.Text := FDQuery1.FieldByName('fullname').AsString;
    EditUsername.Text := FDQuery1.FieldByName('username').AsString;
    EditPassword.Text := FDQuery1.FieldByName('password').AsString;
  end;
end;
procedure TFrmUsers.BtnEditClick(Sender: TObject);
begin
  if (EditUsername.Text = '') then
  begin
    ShowMessage('Masukkan username yang ingin diedit!');
    Exit;
  end;
  FDQuery1.SQL.Text := 'UPDATE users SET fullname = :fullname, password = :password WHERE username = :username';
FDQuery1.ParamByName('fullname').AsString := EditFullname.Text;

FDQuery1.ParamByName('password').AsString := EditPassword.Text;

FDQuery1.ParamByName('username').AsString := EditUsername.Text;
  FDQuery1.ExecSQL;
  FDQuery1.Close;
  FDQuery1.SQL.Text := 'SELECT * FROM users';
  FDQuery1.Open;
  ShowMessage('Data berhasil diperbarui!');
end;
procedure TFrmUsers.BtnHapusClick(Sender: TObject);
begin
  if (EditUsername.Text = '') then
  begin
    ShowMessage('Masukkan username yang ingin dihapus!');
    Exit;
  end;
  FDQuery1.SQL.Text := 'DELETE FROM users WHERE username = :username';

FDQuery1.ParamByName('username').AsString := EditUsername.Text;
  FDQuery1.ExecSQL;
  FDQuery1.Close;
```

```
  FDQuery1.SQL.Text := 'SELECT * FROM users';
  FDQuery1.Open;
  ShowMessage('Data berhasil dihapus!');
end;
procedure TFrmUsers.BtnClearClick(Sender: TObject);
begin
  EditFullname.Clear;
  EditUsername.Clear;
  EditPassword.Clear;
end;
end.
```

**FormUtama.pas**

```
unit FormUtama;
interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.StdCtrls,
  Vcl.Imaging.pngimage;
type
  TFrmUtama = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    imgHome: TImage;
    imgData: TImage;
    imgUsers: TImage;
    imgLogout: TImage;
    Home: TLabel;
    Data: TLabel;
    Users: TLabel;
    Label4: TLabel;
    Label1: TLabel;
    procedure FormShow(Sender: TObject);
    procedure imgHomeClick(Sender: TObject);
    procedure imgDataClick(Sender: TObject);
    procedure imgUsersClick(Sender: TObject);
    procedure imgLogoutClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FrmUtama: TFrmUtama;
implementation
{$R *.dfm}
uses FormLogin, FormHome, FormData, FormUsers;
procedure TFrmUtama.FormShow(Sender: TObject);
var LoginStatus: boolean;
```

```
begin
  frmLogin := TFrmLogin.Create(self);
  frmLogin.ShowModal;
  LoginStatus := frmLogin.LoginStatus;
  if LoginStatus = False then
  begin
    Application.Terminate;
  end;
end;
procedure
TFrmUtama.imgHomeClick(Sender:
TObject);
begin
  TFrmHome.Create(NIL);
end;
procedure
TFrmUtama.imgDataClick(Sender:
TObject);
begin
  TFrmData.Create(NIL);
end;
procedure
TFrmUtama.imgUsersClick(Sender:
TObject);
begin
  TFrmUsers.Create(NIL);
end;
procedure
TFrmUtama.imgLogoutClick(Sender:
TObject);
begin
  if MessageDlg('Yakin ingin logout?',
mtConfirmation, [mbYes, mbNo], 0) =
mrYes then
  Application.Terminate;
end;
end.
```

**Database.js**

```
const sqlite3 =
require("sqlite3").verbose();
const { logMessage } =
require("../utils/logger");
const { parseArgs } =
require("../utils/cli");
const { dbPath } = parseArgs();
// ---------------------------------------------
// ========== DATABASE &
QUERY HELPER ==========
// ---------------------------------------------
// Membuka database SQLite (mode
read-only).
function openDatabase() {
  return new Promise((resolve, reject) =>
{
    const db = new
sqlite3.Database(dbPath,
sqlite3.OPEN_READONLY, (err) => {
      if (err) {
        logMessage(`Error membuka
database: ${err.message}`);
        reject(err);
      } else {
        logMessage(`Terhubung ke
database: ${dbPath}`);
        resolve(db);
```

```
    }
   });
  });
}
// Menjalankan query SQLite (db.all).
function runQuery(db, query, params =
[]) {
  return new Promise((resolve, reject) =>
{
   db.all(query, params, (err, rows) => {
    if (err) reject(err);
    else resolve(rows);
   });
  });
}
// Menutup koneksi database SQLite.
function closeDatabase(db) {
  return new Promise((resolve, reject) =>
{
   db.close((err) => {
    if (err) {
     logMessage(`Error menutup
database: ${err.message}`);
     reject(err);
    } else {
     logMessage("Koneksi database
ditutup.");
     resolve();
    }
   });
  });
}
module.exports = {
  openDatabase,
  runQuery,
  closeDatabase
};
```

**Loans.js**

```
const { openDatabase, runQuery,
closeDatabase } = require("./database");
const { getTodayDate,
getTomorrowDate } =
require("../utils/date");
const { logTable, logMessage } =
require("../utils/logger");
// ---------------------------------------------
// ========== AMBIL DATA
PEMINJAMAN ==========
// ---------------------------------------------
async function
getPeminjamanTerlambat() {
  let db;
  try {
   db = await openDatabase();
   const today = getTodayDate();
   const query = `
    SELECT
     p.id_peminjaman,
     p.tanggal_pinjam,
     p.tanggal_kembali,
     m.nama,
     m.no_hp,
     b.judul,
```

```
      julianday('${today}') - julianday(
        CASE
          WHEN p.tanggal_kembali LIKE '%/%/%'
          THEN substr(p.tanggal_kembali, 7, 4) || '-' ||
               substr(p.tanggal_kembali, 4, 2) || '-' ||
               substr(p.tanggal_kembali, 1, 2)
          ELSE p.tanggal_kembali
        END
      ) AS hari_terlambat
    FROM peminjaman p
    JOIN mahasiswa m ON p.id_mahasiswa = m.id_mahasiswa
    JOIN buku b ON p.id_buku = b.id_buku
    WHERE p.status = 'Dipinjam'
      AND NOT EXISTS (SELECT 1 FROM pengembalian pk WHERE pk.id_peminjaman = p.id_peminjaman)
      AND julianday('${today}') > julianday(
        CASE
          WHEN p.tanggal_kembali LIKE '%/%/%'
          THEN substr(p.tanggal_kembali, 7, 4) || '-' ||
               substr(p.tanggal_kembali, 4, 2) || '-' ||
               substr(p.tanggal_kembali, 1, 2)
          ELSE p.tanggal_kembali
        END
      )
  `;
  const results = await runQuery(db, query);
  if (results.length > 0) {
    logTable("Peminjaman Terlambat", results, [
      { header: "Nama", key: "nama" },
      { header: "Judul", key: "judul" },
      { header: "Tanggal Kembali", key: "tanggal_kembali" },
    ]);
  }
  return results;
 } catch (err) {
  logMessage(`Error peminjaman terlambat: ${err.message}`);
  throw err;
 } finally {
  if (db) await closeDatabase(db);
 }
}
async function getPeminjamanJatuhTempo() {
 let db;
 try {
  db = await openDatabase();
  const today = getTodayDate();
  const query = `
    SELECT
```

```
      p.id_peminjaman,
      p.tanggal_pinjam,
      p.tanggal_kembali,
      m.nama,
      m.no_hp,
      b.judul
    FROM peminjaman p
    JOIN mahasiswa m ON p.id_mahasiswa = m.id_mahasiswa
    JOIN buku b ON p.id_buku = b.id_buku
    WHERE p.status = 'Dipinjam'
      AND NOT EXISTS (SELECT 1 FROM pengembalian pk WHERE pk.id_peminjaman = p.id_peminjaman)
      AND (
        (p.tanggal_kembali LIKE '%/%/%' AND
          (substr(p.tanggal_kembali, 7, 4) || '-' ||
           substr(p.tanggal_kembali, 4, 2) || '-' ||
           substr(p.tanggal_kembali, 1, 2)) = '${today}'
        )
        OR
        (p.tanggal_kembali NOT LIKE '%/%/%' AND p.tanggal_kembali = '${today}')
      )
  `;
  const results = await runQuery(db, query);
  if (results.length > 0) {
    logTable("Peminjaman Jatuh Tempo Hari Ini", results, [
      { header: "Nama", key: "nama" },
      { header: "Judul Buku", key: "judul" },
      { header: "Tanggal Kembali", key: "tanggal_kembali" },
    ]);
  }
  return results;
 } catch (err) {
  logMessage(`Error peminjaman jatuh tempo: ${err.message}`);
  throw err;
 } finally {
  if (db) await closeDatabase(db);
 }
}
async function getPeminjamanAkanJatuhTempoBesok() {
 let db;
 try {
  db = await openDatabase();
  const tomorrow = getTomorrowDate();
  const query = `
    SELECT
      p.id_peminjaman,
      p.tanggal_pinjam,
      p.tanggal_kembali,
      m.nama,
```

```
      m.no_hp,
      b.judul
    FROM peminjaman p
    JOIN mahasiswa m ON p.id_mahasiswa = m.id_mahasiswa
    JOIN buku b ON p.id_buku = b.id_buku
    WHERE p.status = 'Dipinjam'
      AND NOT EXISTS (SELECT 1 FROM pengembalian pk WHERE pk.id_peminjaman = p.id_peminjaman)
      AND (
        (p.tanggal_kembali LIKE '%/%/%' AND
          (substr(p.tanggal_kembali, 7, 4) || '-' ||
           substr(p.tanggal_kembali, 4, 2) || '-' ||
           substr(p.tanggal_kembali, 1, 2)) = '${tomorrow}'
        )
        OR
        (p.tanggal_kembali NOT LIKE '%/%/%' AND p.tanggal_kembali = '${tomorrow}')
      )
  `;
  const results = await runQuery(db, query);
  if (results.length > 0) {
    logTable("Pengembalian Buku Akan Jatuh Tempo Besok", results, [
      { header: "Nama", key: "nama" },
      { header: "Judul Buku", key: "judul" },
      { header: "Tanggal Kembali", key: "tanggal_kembali" },
    ]);
  }
  return results;
 } catch (err) {
  logMessage(`Error Pengembalian Buku akan jatuh tempo besok: ${err.message}`);
  throw err;
 } finally {
  if (db) await closeDatabase(db);
 }
}
async function getAllActiveLoans() {
 let db;
 try {
  db = await openDatabase();
  const today = getTodayDate();
  const tomorrow = getTomorrowDate();
  const query = `
   WITH peminjaman_iso AS (
    SELECT
     p.id_peminjaman,
     p.tanggal_pinjam,
     p.tanggal_kembali,
     p.status,
     m.nama,
```

```
        m.no_hp,
        b.judul,
        CASE
          WHEN p.tanggal_kembali LIKE
'%/%/%'
          THEN substr(p.tanggal_kembali,
7, 4) || '-' ||
               substr(p.tanggal_kembali, 4,
2) || '-' ||
               substr(p.tanggal_kembali, 1,
2)
          ELSE p.tanggal_kembali
        END AS tanggal_kembali_iso
      FROM peminjaman p
      JOIN mahasiswa m ON
p.id_mahasiswa = m.id_mahasiswa
      JOIN buku b ON p.id_buku =
b.id_buku
      WHERE
        p.status = 'Dipinjam'
        AND NOT EXISTS (
          SELECT 1 FROM pengembalian
pk
          WHERE pk.id_peminjaman =
p.id_peminjaman
        )
    )
    SELECT
      id_peminjaman,
      tanggal_pinjam,
      tanggal_kembali,
      status,
      nama,
      no_hp,
      judul,
      tanggal_kembali_iso,
      CASE
        WHEN julianday('${today}') >
julianday(tanggal_kembali_iso)
        THEN 'Terlambat'
        WHEN tanggal_kembali_iso =
'${today}'
        THEN 'Jatuh Tempo Hari Ini'
        WHEN tanggal_kembali_iso =
'${tomorrow}'
        THEN 'Akan Jatuh Tempo Besok'
        ELSE 'Aktif'
      END AS status_peminjaman
    FROM peminjaman_iso
    ORDER BY tanggal_kembali_iso
  `;
  return await runQuery(db, query);
 } catch (err) {
  throw err;
 } finally {
  if (db) await closeDatabase(db);
 }
}
module.exports = {
 getPeminjamanTerlambat,
 getPeminjamanJatuhTempo,
 getPeminjamanAkanJatuhTempoBesok
,
 getAllActiveLoans
```

```
};
```

**Notifications.js**

```
const cron = require("node-cron");

const { logMessage } =
require("../utils/logger");

const { formatInterval } =
require("../utils/formatter");

const { parseArgs } =
require("../utils/cli");

const {

  getPeminjamanTerlambat,

  getPeminjamanJatuhTempo,

  getPeminjamanAkanJatuhTempoBesok

} = require("./loans");

const { checkWhatsAppConnection,
sendWhatsAppMessage } =
require("./whatsapp");

// ----------------------------------------------
-

// ========== FUNGSI
NOTIFIKASI WA ==========

// ----------------------------------------------
-

async function
kirimNotifikasiTerlambat() {

  try {

    if (!checkWhatsAppConnection()) {

      logMessage("WhatsApp tidak
terhubung. Notifikasi terlambat
dibatalkan.");

      return;

    }

    const dataTerlambat = await
getPeminjamanTerlambat();

    for (const row of dataTerlambat) {

      const { nama, no_hp, judul,
hari_terlambat, tanggal_kembali } =
row;

      if (!no_hp) continue;

      const denda =
Math.ceil(hari_terlambat) * 1000;

      const pesan = `*NOTIFIKASI
KETERLAMBATAN*\n\nHalo
${nama},\nBuku "${judul}" terlambat
dikembalikan (jatuh tempo:
${tanggal_kembali}).\nTerlambat:
${Math.ceil(

        hari_terlambat

      )} hari\nDenda: Rp
${denda.toLocaleString("id-
ID")}\n\nMohon segera dikembalikan.
Terima kasih.`;

      try {

        await
sendWhatsAppMessage(no_hp, pesan);

      } catch (err) {

        logMessage(`Gagal kirim notifikasi
terlambat ke ${no_hp}:
${err.message}`);

      }

    }

  } catch (err) {

    logMessage(`Error notifikasi
terlambat: ${err.message}`);

  }

}

async function
kirimNotifikasiJatuhTempoHariIni() {

  try {

    if (!checkWhatsAppConnection()) {
```

```
      logMessage("WhatsApp tidak terhubung. Notifikasi hari ini dibatalkan.");
      return;
    }
    const dataHariIni = await getPeminjamanJatuhTempo();
    for (const row of dataHariIni) {
      const { nama, no_hp, judul, tanggal_kembali } = row;
      if (!no_hp) continue;
      const pesan = `*PENGINGAT JATUH TEMPO*\n\nHalo ${nama},\nBuku "${judul}" jatuh tempo hari ini (${tanggal_kembali}).\nSegera kembalikan untuk menghindari denda.`;
      try {
        await sendWhatsAppMessage(no_hp, pesan);
      } catch (err) {
        logMessage(`Gagal kirim notifikasi hari ini ke ${no_hp}: ${err.message}`);
      }
    }
  } catch (err) {
    logMessage(`Error notifikasi hari ini: ${err.message}`);
  }
}
async function kirimNotifikasiJatuhTempoBesok() {
  try {
    if (!checkWhatsAppConnection()) {
      logMessage("WhatsApp tidak terhubung. Notifikasi besok dibatalkan.");
      return;
    }
    const dataBesok = await getPeminjamanAkanJatuhTempoBesok();
    for (const row of dataBesok) {
      const { nama, no_hp, judul, tanggal_kembali } = row;
      if (!no_hp) continue;
      const pesan = `*PENGINGAT JATUH TEMPO*\n\nHalo ${nama},\nBuku "${judul}" akan jatuh tempo besok (${tanggal_kembali}).\nHarap siapkan buku untuk dikembalikan.`;
      try {
        await sendWhatsAppMessage(no_hp, pesan);
      } catch (err) {
        logMessage(`Gagal kirim notifikasi besok ke ${no_hp}: ${err.message}`);
      }
    }
  } catch (err) {
    logMessage(`Error notifikasi besok: ${err.message}`);
  }
}
async function manualSendNotifications() {
  logMessage("Mulai kirim notifikasi manual...");
```

```
  const terlambatData = await
getPeminjamanTerlambat();

  const hariIniData = await
getPeminjamanJatuhTempo();

  const besokData = await
getPeminjamanAkanJatuhTempoBesok()
;

  if (terlambatData.length > 0) await
kirimNotifikasiTerlambat();

  if (hariIniData.length > 0) await
kirimNotifikasiJatuhTempoHariIni();

  if (besokData.length > 0) await
kirimNotifikasiJatuhTempoBesok();

  return {

    status: "success",

    message: "Notifikasi berhasil
dikirim",

    detail: {

      terlambat: terlambatData.length,

      hari_ini: hariIniData.length,

      besok: besokData.length,

    },

  };

}

// ---------------------------------------------

// ========== PENJADWALAN
CRON JOB ==========

// ---------------------------------------------

async function sendNotifications() {

  logMessage("Mengirim notifikasi
berdasarkan data terbaru...");

  await kirimNotifikasiTerlambat();

  await
kirimNotifikasiJatuhTempoHariIni();

  await
kirimNotifikasiJatuhTempoBesok();

  logMessage("Selesai mengirim
notifikasi.");

}

function
setupNotificationSchedule(client) {

  const { intervalHari, intervalJam,
intervalMenit, totalMenit,
generateCronFromInterval } =
parseArgs();

  logMessage("WhatsApp Web siap.
Memulai penjadwalan notifikasi...");

  setTimeout(async () => {

    logMessage("Menjalankan notifikasi
pertama kali...");

    await sendNotifications();

  }, 3000);

  if (totalMenit === 0) {

    logMessage("Mode sekali jalan. Tidak
ada penjadwalan berikutnya.");

  } else {

    const cronSchedule =
generateCronFromInterval(totalMenit,
intervalHari, intervalJam,
intervalMenit);

    cron.schedule(cronSchedule, async ()
=> {

      logMessage(`Cron job: Kirim
notifikasi (interval
${formatInterval(intervalHari,
intervalJam, intervalMenit)})`);

      await sendNotifications();

    });

    logMessage(`Penjadwalan berikutnya
diatur dengan interval
```

```
${formatInterval(intervalHari, intervalJam, intervalMenit)}`);
  }
}
module.exports = {
  kirimNotifikasiTerlambat,
  kirimNotifikasiJatuhTempoHariIni,
  kirimNotifikasiJatuhTempoBesok,
  manualSendNotifications,
  setupNotificationSchedule,
  sendNotifications
};
```

**Whatsapp.js**

```
const fs = require("fs");
const path = require("path");
const qrcode = require("qrcode-terminal");
const chalk = require("chalk");
const { Client, LocalAuth } = require("whatsapp-web.js");
const { logMessage } = require("../utils/logger");
// ----------------------------------------
// ========== KONFIGURASI WA WEB ==========
// ----------------------------------------
const client = new Client({
  authStrategy: new LocalAuth(),
  puppeteer: {
    headless: true,
    executablePath: process.env.PUPPETEER_EXECUTABLE_PATH,
    args: [
      "--no-sandbox",
      "--disable-setuid-sandbox",
      "--disable-dev-shm-usage",
      "--disable-accelerated-2d-canvas",
      "--no-first-run",
      "--no-zygote",
      "--single-process",
      "--disable-gpu",
      "--disable-extensions",
      "--disable-features=site-per-process",
    ],
  },
});
// Event: Tampilkan QR code jika sesi belum ada.
client.on("qr", (qr) => {
  console.log(chalk.yellow("\nScan QR Code ini untuk login:"));
  qrcode.generate(qr, { small: true });
  console.log(chalk.cyan("\nSilakan scan QR di atas dengan WhatsApp di ponsel Anda."));
});
// Event: Terkoneksi.
client.on("ready", () => {
  logMessage("WhatsApp Web siap digunakan!");
```

```
  console.log(chalk.cyan("\nWhatsApp terhubung!"));

  // Tampilkan menu utama setelah WhatsApp terhubung

  const { showMainMenu } = require('../ui/menu');

  showMainMenu();

  // Setup notifikasi setelah WhatsApp siap (dengan process.nextTick untuk menghindari circular dependency)

  process.nextTick(() => {

    const { setupNotificationSchedule } = require('./notifications');

    setupNotificationSchedule(client);

  });

});

// Event: Pesan masuk.

client.on("message", async (msg) => {

  logMessage(`Pesan dari ${msg.from}: ${msg.body}`);

});

// Event: Terputus.

client.on("disconnected", (reason) => {

  logMessage(`WhatsApp terputus: ${reason}`);

  console.log(chalk.red("\nWhatsApp terputus! Mencoba menghubungkan kembali..."));

  setTimeout(() => client.initialize(), 5000);

});

// Event: Autentikasi gagal.

client.on("auth_failure", (msg) => {

  logMessage(`Autentikasi gagal: ${msg}`);

  console.log(chalk.red("\nAutentikasi WhatsApp gagal! Silakan coba lagi."));

});

// Inisialisasi client WhatsApp dengan penanganan error.

async function initializeWhatsApp() {

  try {

    console.log(chalk.cyan("\nMenginisialisasi WhatsApp..."));

    await client.initialize();

  } catch (err) {

    logMessage(`Error inisialisasi: ${err.message}`);

    console.log(chalk.red(`\nGagal menghubungkan WhatsApp: ${err.message}`));

    if (

      err.message.includes("Protocol error") ||

      err.message.includes("Execution context was destroyed")

    ) {

      console.log(chalk.yellow("\nAuth error—bersihkan session dan coba ulang…"));

      try {

        // 1) Tutup Puppeteer

        await client.destroy();

      } catch { }

      // 2) Hapus folder session dengan helper baru

      await clearSessionFolder();
```

```
      // 3) Coba inisialisasi ulang
      try {
        await client.initialize();
      } catch (retryErr) {
        logMessage(`Error retry init: ${retryErr.message}`);
        console.log(chalk.red(`\nGagal konek ulang: ${retryErr.message}`));
        setTimeout(() => process.exit(1), 5000);
      }
    } else {
      console.log(chalk.yellow("\nCoba ulang 5 detik…"));
      setTimeout(() => initializeWhatsApp(), 5000);
    }
  }
}

// Hapus Session Web WhatsApp jika ada error yang berkaitan dengan sesi.
async function clearSessionFolder() {
  const sessionFolder = path.join(process.cwd(), ".wwebjs_auth");
  const maxRetries = 5;
  for (let i = 0; i < maxRetries; i++) {
    try {
      if (fs.existsSync(sessionFolder)) {
        await fs.promises.rm(sessionFolder, { recursive: true, force: true });
        console.log("Session folder berhasil dihapus.");
      }
      return;
    } catch (err) {
      if (err.code === "EPERM" && i < maxRetries - 1) {
        console.warn(`EPERM saat hapus session (percobaan ${i + 1}), retry 200ms…`);
        await new Promise((r) => setTimeout(r, 200));
        continue;
      }
      console.error("Gagal hapus session folder:", err);
      return;
    }
  }
}

// Mengirim pesan WhatsApp ke nomor tertentu.
async function sendWhatsAppMessage(phone, message) {
  if (!phone || !message) throw new Error("Nomor dan pesan harus diisi!");
  logMessage(`Mengirim pesan ke ${phone}`);
  const formattedNumber = phone.replace(/\D/g, "") + "@c.us";
  const isRegistered = await client.isRegisteredUser(formattedNumber);
  if (!isRegistered) {
    logMessage(`Nomor ${phone} tidak terdaftar di WhatsApp.`);
```

```
    throw new Error("Nomor tidak terdaftar di WhatsApp!");
  }
  await client.sendMessage(formattedNumber, message);
  logMessage(`Pesan terkirim ke ${phone}`);
  return true;
}
// Mengecek koneksi WA. True jika info klien tersedia.
function checkWhatsAppConnection() {
  try {
    return client.info && client.info.wid ? true : false;
  } catch {
    return false;
  }
}
module.exports = {
  client,
  initializeWhatsApp,
  sendWhatsAppMessage,
  checkWhatsAppConnection
};
```

**Menu.js**

```
const readline = require("readline");
const chalk = require("chalk");
const clear = require("clear");
const { displayTable } = require("./table");
const {
  getPeminjamanTerlambat,
  getPeminjamanJatuhTempo,
  getPeminjamanAkanJatuhTempoBesok,
  getAllActiveLoans
} = require("../services/loans");
const { manualSendNotifications } = require("../services/notifications");
const { formatInterval } = require("../utils/formatter");
const { parseArgs } = require("../utils/cli");
const { formatRupiah } = require("../utils/formatter");
// ----------------------------------------
// ========== INISIALISASI CLI ==========
// ----------------------------------------
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});
// Tampilkan menu utama di CLI
function showMainMenu() {
  clear();
  console.log(chalk.cyan("========== MENU UTAMA =========="));
  console.log("1. Cek Pengembalian Buku Terlambat");
  console.log("2. Cek Pengembalian Buku Akan Jatuh Tempo Besok");
```

```
  console.log("3. Cek Pengembalian Buku Jatuh Tempo Hari Ini");
  console.log("4. Kirim Notifikasi Manual");
  console.log("5. Lihat Semua Pengembalian Buku Aktif");
  console.log("6. Lihat Jadwal Notifikasi Otomatis");
  console.log("7. Keluar");
  rl.question(chalk.green("\nPilih menu (1-7): "), handleMenuChoice);
}

// Handle pilihan menu
async function handleMenuChoice(choice) {
  switch (choice) {
    case "1":
      await showTerlambat();
      break;
    case "2":
      await showAkanJatuhTempoBesok();
      break;
    case "3":
      await showJatuhTempoHariIni();
      break;
    case "4":
      await sendNotifications();
      break;
    case "5":
      await showAllLoans();
      break;
    case "6":
      await showJadwal();
      break;
    case "7":
      exitApp();
      break;
    default:
      console.log(chalk.red("\nPilihan tidak valid!"));
      pressAnyKeyToContinue();
  }
}

// ----------------------------------------
// ============ FUNGSI UTILITAS ============
// ----------------------------------------
function pressAnyKeyToContinue() {
  console.log(chalk.cyan("\nTekan Enter untuk kembali ke menu utama..."));
  rl.question("", showMainMenu);
}
function exitApp() {
  console.log(chalk.yellow("\nTerima kasih telah menggunakan aplikasi!"));
  rl.close();
  process.exit(0);
}

// ----------------------------------------
// ========== FUNGSI TAMPILAN DATA =========
// ----------------------------------------
```

```
// Tampilkan Pengembalian Buku terlambat
async function showTerlambat() {
  clear();
  console.log(chalk.cyan("\n=== DAFTAR PENGEMBALIAN BUKU TERLAMBAT ==="));
  try {
    const data = await getPeminjamanTerlambat();

    if (data.length > 0) {
      displayTable("Pengembalian Buku Terlambat", data, [
        { header: "Nama", key: "nama" },
        { header: "Judul Buku", key: "judul" },
        {
          header: "Terlambat (Hari)",
          key: "hari_terlambat",
          formatter: val => Math.ceil(val)
        },
        { header: "Tanggal Kembali", key: "tanggal_kembali" },
        {
          header: "Denda",
          key: "hari_terlambat",
          formatter: val => formatRupiah(Math.ceil(val) * 1000)
        }
      ]);
    } else {
      console.log(chalk.green("\nTidak ada pengembalian buku terlambat"));
    }
  } catch (error) {
    handleError(error);
  }
  pressAnyKeyToContinue();
}
// Tampilkan pengembalian buku akan jatuh tempo besok
async function showAkanJatuhTempoBesok() {
  clear();
  console.log(chalk.cyan("\n=== DAFTAR PENGEMBALIAN BUKU AKAN JATUH TEMPO BESOK ==="));

  try {
    const data = await getPeminjamanAkanJatuhTempoBesok();
    if (data.length > 0) {
      displayTable("Akan Jatuh Tempo Besok", data, [
        { header: "Nama", key: "nama" },
        { header: "Judul Buku", key: "judul" },
        { header: "Tanggal Kembali", key: "tanggal_kembali" },
        { header: "No. HP", key: "no_hp" }
      ]);
    } else {
```

```
      console.log(chalk.green("\nTidak ada pengembalian buku yang akan jatuh tempo besok"));
    }
  } catch (error) {
    handleError(error);
  }
  pressAnyKeyToContinue();
}
// Tampilkan pengembalian buku jatuh tempo hari ini
async function showJatuhTempoHariIni() {
  clear();
  console.log(chalk.cyan("\n=== DAFTAR PENGEMBALIAN BUKU JATUH TEMPO HARI INI ==="));
  try {
    const data = await getPeminjamanJatuhTempo();

    if (data.length > 0) {
      displayTable("Jatuh Tempo Hari Ini", data, [
        { header: "Nama", key: "nama" },
        { header: "Judul Buku", key: "judul" },
        { header: "Tanggal Kembali", key: "tanggal_kembali" },
        { header: "No. HP", key: "no_hp" }
      ]);
    } else {
      console.log(chalk.green("\nTidak ada pengembalian buku jatuh tempo hari ini"));
    }
  } catch (error) {
    handleError(error);
  }
  pressAnyKeyToContinue();
}
// Tampilkan semua pengembalian buku aktif
async function showAllLoans() {
  clear();
  console.log(chalk.cyan("\n=== SEMUA PENGEMBALIAN BUKU AKTIF ==="));
  try {
    const data = await getAllActiveLoans();
    if (data.length > 0) {
      displayTable("Pengembalian Aktif", data, [
        { header: "Nama", key: "nama" },
        { header: "Judul Buku", key: "judul" },
        { header: "Tanggal Pinjam", key: "tanggal_pinjam" },
        { header: "Tanggal Kembali", key: "tanggal_kembali" },
        { header: "Status", key: "status_peminjaman" }
      ]);
    } else {
```

```
      console.log(chalk.green("\nTidak ada pengembalian buku aktif"));
    }
  } catch (error) {
    handleError(error);
  }
  pressAnyKeyToContinue();
}
// ----------------------------------------
// ========= FUNGSI NOTIFIKASI ============
// ----------------------------------------
// Kirim notifikasi manual
async function sendNotifications() {
  clear();
  console.log(chalk.cyan("\n=== KIRIM NOTIFIKASI MANUAL ==="));
  try {
    const [terlambat, hariIni, besok] = await Promise.all([
      getPeminjamanTerlambat(),
      getPeminjamanJatuhTempo(),
      getPeminjamanAkanJatuhTempoBesok()
    ]);
    console.log(chalk.yellow("\nStatus Pengembalian Buku:"));
    console.log(`- Pengembalian Buku Terlambat: ${terlambat.length}`);
    console.log(`- Jatuh Tempo Hari Ini: ${hariIni.length}`);
    console.log(`- Akan Jatuh Tempo Besok: ${besok.length}`);
    if (terlambat.length + hariIni.length + besok.length === 0) {
      console.log(chalk.green("\nTidak ada notifikasi yang perlu dikirim"));
      pressAnyKeyToContinue();
      return;
    }
    rl.question(chalk.yellow("\nKirim semua notifikasi? (y/n) "), async (answer) => {
      if (answer.toLowerCase() === 'y') {
        console.log(chalk.yellow("\nMengirim notifikasi..."));
        try {
          const result = await manualSendNotifications();
          console.log(chalk.green("\nNotifikasi berhasil dikirim:"));
          console.log(`- Terlambat: ${result.detail.terlambat}`);
          console.log(`- Hari Ini: ${result.detail.hari_ini}`);
          console.log(`- Besok: ${result.detail.besok}`);
        } catch (error) {
          handleError(error);
        }
      } else {
        console.log(chalk.yellow("\nPengiriman notifikasi dibatalkan"));
      }
      pressAnyKeyToContinue();
    });
  } catch (error) {
```

```
    handleError(error);
  }
}

// ----------------------------------------
// ========== FUNGSI JADWAL ================
// ----------------------------------------
// Tampilkan jadwal notifikasi
async function showJadwal() {
  clear();
  console.log(chalk.cyan("\n=== JADWAL NOTIFIKASI OTOMATIS ==="));
  const {
    intervalHari,
    intervalJam,
    intervalMenit,
    totalMenit
  } = parseArgs();
  console.log(chalk.yellow("\nKonfigurasi Saat Ini:"));
  console.log(`- Interval: ${formatInterval(intervalHari, intervalJam, intervalMenit)}`);
  console.log(`- Mode: ${totalMenit === 0 ? 'Sekali Jalankan' : 'Berulang'}`);
  console.log(`- Waktu Sistem: ${new Date().toLocaleString('id-ID')}`);
  try {
    const [terlambat, hariIni, besok] = await Promise.all([
      getPeminjamanTerlambat(),
      getPeminjamanJatuhTempo(),
      getPeminjamanAkanJatuhTempoBesok()
    ]);
    console.log(chalk.yellow("\nStatus Terkini:"));
    console.log(`- Pengembalian Buku Terlambat: ${terlambat.length}`);
    console.log(`- Jatuh Tempo Hari Ini: ${hariIni.length}`);
    console.log(`- Akan Jatuh Tempo Besok: ${besok.length}`);
  } catch (error) {
    handleError(error);
  }
  pressAnyKeyToContinue();
}
// ----------------------------------------
// ========== ERROR HANDLING ===============
// ----------------------------------------
function handleError(error) {
  console.log(chalk.red("\nTerjadi Kesalahan:"));
  console.log(chalk.red(`- Pesan: ${error.message}`));
  console.log(chalk.red(`- Stack: ${error.stack}`));
}
module.exports = {
  showMainMenu
};
```

**Table,js**

```
const chalk = require("chalk");

const Table = require("cli-table3");

// ----------------------------------------

// ========== TAMPIL DI KONSOL ==========

// ----------------------------------------

// Menampilkan data dalam bentuk tabel di CLI menggunakan cli-table3.

function displayTable(title, data, columns) {

  const table = new Table({

    head: columns.map((col) => chalk.cyan(col.header)),

    chars: {

      top: "═",

      "top-mid": "╤",

      "top-left": "╔",

      "top-right": "╗",

      bottom: "═",

      "bottom-mid": "╧",

      "bottom-left": "╚",

      "bottom-right": "╝",

      left: "║",

      "left-mid": "╟",

      mid: "─",

      "mid-mid": "┼",

      right: "║",

      "right-mid": "╢",

      middle: "│",

    },

  });

  data.forEach((item) => {

    const row = columns.map((col) => {

      const value = item[col.key] || "";

      return col.formatter ? col.formatter(value) : value;

    });

    table.push(row);

  });

  console.log(chalk.bold(`\n${title}:`));

  console.log(table.toString());

  return table;

}

module.exports = {

  displayTable

};
```

**Cli.js**

```
const argparse = require("argparse");

const { consoleLog } = require("./logger");

// ----------------------------------------

// ========== PARSE ARGUMENT CLI ==========

// ----------------------------------------

function parseArgs() {

  const parser = new argparse.ArgumentParser({

    description: "WhatsApp API dengan dukungan database SQLite",

  });

  parser.add_argument("--database", {

    help: "Path ke database SQLite",
```

```
    default: "G:\\My Drive\\Development\\Perpus\\db_perpus.db", // Sesuaikan default path
  });
  parser.add_argument("--interval", {
    help: "Interval pengiriman notifikasi (format: 1d, 1h, 1m, 1d:2h:30m, dst). Default: 24 jam",
    default: "24h",
  });
  const args = parser.parse_args();
  const dbPath = args.database;
  const intervalData = parseInterval(args.interval);

  let intervalHari = 0;
  let intervalJam = 24; // default 24 jam
  let intervalMenit = 0;
  let totalMenit = 24 * 60; // default 24 jam
  if (!intervalData) {
    consoleLog(`Interval tidak valid: ${args.interval}. Gunakan default 24 jam.`);
  } else if (intervalData.totalMinutes <= 0) {
    consoleLog(`Interval <= 0: ${args.interval}. Gunakan default 24 jam.`);
  } else {
    intervalHari = intervalData.days;
    intervalJam = intervalData.hours;
    intervalMenit = intervalData.minutes;
    totalMenit = intervalData.totalMinutes;
    if (totalMenit === 0) consoleLog("Interval 0: Notifikasi hanya sekali saat start.");
  }
  return {
    dbPath,
    intervalHari,
    intervalJam,
    intervalMenit,
    totalMenit,
    generateCronFromInterval
  };
}
// Memparsing string interval (misal: "24", "1d", "1h", "1d:2h:30m") menjadi {days, hours, minutes, totalMinutes}.
function parseInterval(intervalStr) {
  // Jika hanya angka, anggap jam
  if (/^\d+$/.test(intervalStr)) {
    const val = parseInt(intervalStr, 10);
    const days = Math.floor(val / 24);
    const hours = val % 24;
    return {
      days: days,
      hours: hours,
      minutes: 0,
      totalMinutes: val * 60
    };
  }
```

```
  // Cek pola sederhana: 1d, 1h, 1m
  const simplePattern = /^(\d+)([dhm])$/i;
  const simpleMatch = intervalStr.match(simplePattern);
  if (simpleMatch) {
    const value = parseInt(simpleMatch[1], 10);
    const unit = simpleMatch[2].toLowerCase();
    let d = 0,
      h = 0,
      m = 0;
    if (unit === "d") d = value;
    else if (unit === "h") h = value;
    else if (unit === "m") m = value;
    return {
      days: d,
      hours: h,
      minutes: m,
      totalMinutes: d * 1440 + h * 60 + m,
    };
  }
  return null;
}

// ----------------------------------------
// ========== KONVERSI CRON JOB ==========
// ----------------------------------------

// Membuat ekspresi cron dari totalMenit, hari, jam, dan menit.
function generateCronFromInterval(total, hari, jam, menit) {
  if (total === 0) return null;
  // Validasi jam (harus 0-23)
  if (jam >= 24) {
    hari = hari + Math.floor(jam / 24);
    jam = jam % 24;
  }
  // Interval < 1 jam
  if (total < 60) return `*/${menit} * * * *`; // Setiap N menit
  // Interval < 1 hari
  if (total < 1440) {
    if (menit === 0) return `0 */${jam} * * *`; // N jam tepat
    return `${menit} */${jam} * * *`;
  }
  // Kelipatan hari pas (00:00)
  if (menit === 0 && jam === 0) return `0 0 */${hari} * *`;
  // Kombinasi hari, jam, menit
  return `${menit} ${jam} */${hari} * *`;
}

module.exports = {
  parseArgs
};
```

**Date.js**

```
// ----------------------------------------
// ========== FUNGSI PEMBANTU TANGGAL ==========
```

```
// ---------------------------------------------

function getTodayDate() {

  const today = new Date();

  const y = today.getFullYear();

  const m = String(today.getMonth() +
1).padStart(2, "0");

  const d =
String(today.getDate()).padStart(2, "0");

  return `${y}-${m}-${d}`;

}

function getTomorrowDate() {

  const tomorrow = new Date();

  tomorrow.setDate(tomorrow.getDate()
+ 1);

  const y = tomorrow.getFullYear();

  const m = String(tomorrow.getMonth()
+ 1).padStart(2, "0");

  const d =
String(tomorrow.getDate()).padStart(2,
"0");

  return `${y}-${m}-${d}`;

}

// Tambahkan fungsi untuk mendapatkan
tanggal dalam format DD/MM/YYYY

function
getFormattedDueDate(dateString) {

  const [year, month, day] =
dateString.split('-');

  return `${day}/${month}/${year}`;

}

module.exports = {

  getTodayDate,

  getTomorrowDate,

  getFormattedDueDate

};
```

**Formatter.js**

```
// Format angka menjadi string Rupiah.

function formatRupiah(angka) {

  return new Intl.NumberFormat("id-ID",
{

    style: "currency",

    currency: "IDR",

    minimumFractionDigits: 0,

  }).format(angka);

}

// Memformat interval (hari, jam, menit)
menjadi string yang mudah dibaca.

function formatInterval(hari, jam, menit)
{

  const parts = [];

  if (hari > 0) parts.push(`${hari} hari`);

  if (jam > 0) parts.push(`${jam} jam`);

  if (menit > 0) parts.push(`${menit}
menit`);

  return parts.length > 0 ? parts.join(" ") :
"0 (sekali jalan)";

}

module.exports = {

  formatRupiah,

  formatInterval

};
```

**Logger.js**

```
const fs = require("fs");

const path = require("path");
```

```
const Table = require("cli-table3");
const chalk = require("chalk");
// ---------------------------------------
// ========== BAGIAN LOGGING ==========
// ---------------------------------------
// Memastikan direktori log tersedia. Jika belum ada, maka dibuat.
function ensureLogDir() {
  const logDir = path.join(__dirname, "../logs");
  if (!fs.existsSync(logDir)) fs.mkdirSync(logDir);
  return logDir;
}
// Menghasilkan nama file log berdasarkan tanggal saat ini (format: log_YYYY-MM-DD.txt).
function getLogFileName() {
  const now = new Date();
  const year = now.getFullYear();
  const month = String(now.getMonth() + 1).padStart(2, "0");
  const day = String(now.getDate()).padStart(2, "0");
  return path.join(ensureLogDir(), `log_${year}-${month}-${day}.txt`);
}
// Menyimpan pesan log ke file log (mode append). Tidak ditampilkan di konsol.
function logMessage(msg, type = "info") {
  const timestamp = new Date().toISOString();
  fs.appendFileSync(getLogFileName(), `[${timestamp}] ${msg}\n`);
}
// Log versi tabel ke file log (untuk menulis data yang lebih terstruktur).
function logTable(title, data, columns) {
  const table = new Table({
    head: columns.map((col) => chalk.cyan(col.header)),
    chars: {
      top: "═",
      "top-mid": "╤",
      "top-left": "╔",
      "top-right": "╗",
      bottom: "═",
      "bottom-mid": "╧",
      "bottom-left": "╚",
      "bottom-right": "╝",
      left: "║",
      "left-mid": "╟",
      mid: "─",
      "mid-mid": "┼",
      right: "║",
      "right-mid": "╢",
      middle: "│",
    },
  });
  data.forEach((item) => {
```

```
    const row = columns.map((col) =>
(item[col.key] || "").toString());
    table.push(row);
  });
  // Tulis versi ringkas ke file log
  fs.appendFileSync(
    getLogFileName(),
    `\n[${new Date().toISOString()}]
${title}:\n` +
      data
        .map((item) =>
          columns.map((col) =>
`${col.header}: ${item[col.key] ||
""}`).join(" | ")
        )
        .join("\n") +
      "\n\n"
  );
  return table;
}
// Fungsi log sederhana sebelum
logMessage tersedia (tidak ditampilkan
di konsol, hanya ke file).
function consoleLog(msg) {
  fs.appendFileSync(getLogFileName(),
`[${new Date().toISOString()}]
${msg}\n`);
}
module.exports = {
  ensureLogDir,
  getLogFileName,
  logMessage,
  logTable,
  consoleLog
};
```

**App.js**

```
const fs = require("fs");
const path = require("path");
const chalk = require("chalk");
const clear = require("clear");
const { parseArgs } =
require("./utils/cli");
const { showMainMenu } =
require("./ui/menu");
const { initializeWhatsApp } =
require("./services/whatsapp");
const { logMessage, logTable } =
require("./utils/logger");
const { formatInterval } =
require("./utils/formatter");
// Fungsi utama untuk menginisialisasi
aplikasi
function initializeApp() {
  // Parse argumen CLI
  const { dbPath, intervalHari,
intervalJam, intervalMenit, totalMenit }
= parseArgs();
  console.log(chalk.cyan("\n===
INFORMASI APLIKASI ==="));
  console.log(`Database:
${chalk.green(dbPath)}`);
  console.log(
    `Interval Notifikasi: ${chalk.green(
      totalMenit === 0 ? "Sekali jalan" :
formatInterval(intervalHari, intervalJam,
intervalMenit)
    )}`
```

```
  );
  logTable("Informasi Aplikasi", [
    {
      name: "WhatsApp API",
      status: "Menunggu koneksi",
      database: dbPath,
      interval: totalMenit === 0 ? "Sekali jalan" : formatInterval(intervalHari, intervalJam, intervalMenit),
    },
  ], [
    { header: "Nama", key: "name" },
    { header: "Status", key: "status" },
    { header: "Database", key: "database" },
    { header: "interval", key: "interval" },
    { header: "Jadwal", key: "jadwal" },
  ]);
  console.log(chalk.cyan("\nMenunggu WhatsApp terhubung..."));
  // Inisialisasi WhatsApp
  // Menu utama akan ditampilkan hanya setelah WhatsApp terhubung
  initializeWhatsApp();
}
initializeApp();
```

# KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

## PROPOSAL

| | |
|---|---|
| **Mahasiswa : ROSDIANA** | **Pembimbing I : Muh. Basri, ST., MT.** |
| **NIM : 220280043** | **Pembimbing II : : Marlina, S.Kom.,M.Kom** |
| **Judul Skripsi : PENGEMBANGAN SISTEM INFORMASI PERPUSTAKAAN MENGGUNAKAN SISTEM BARCODE** | |

| ARAHAN PEMBIMBING I | HARI/TGL & PARAF PEMBIMBING | ARAHAN PEMBIMBING II | HARI/TGL & PARAF PEMBIMBING |
|---|---|---|---|
| Konsultasi 1 | | Konsultasi 1<br>Cari penelitian sebelumnya yg ada di UMPAR yg akan dikembangkan. | 28/2/2024 |
| Konsultasi 2 | | Konsultasi 2<br>Cari penelitian yg ada di UMPAR kemudian di analisa apa yg ingin dikembangkan | 15/03/2024 |
| Konsultasi 3 | | Konsultasi 3 | |
| Konsultasi 4<br>Bahan ke bab berikutnya | | Konsultasi 4<br>Acc Ujian Proposal | |
| Konsultasi 5 | | Konsultasi 5 | |

*Lanjut ke halaman sebelah...*

**Perhatian :**

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

*Lanjutan ...*

| ARAHAN PEMBIMBING I | HARI/TGL & PARAF PEMBIMBING | ARAHAN PEMBIMBING II | HARI/TGL & PARAF PEMBIMBING |
|---|---|---|---|
| Konsultasi 6 | | Konsultasi 6 | |
| Konsultasi 7 | | Konsultasi 7 | |
| Konsultasi 8 | | Konsultasi 8 | |
| Konsultasi 9 | | Konsultasi 9 | |
| Konsultasi 10 | | Konsultasi 10 | |

Parepare, 28 Februari 2024

Mengetahui
Ketua Program Studi

**Marlina, S.Kom.,M.Kom.**
NBM. 1162 680

Mahasiswa

**ROSDIANA**
**NIM. 220280043**

**Perhatian :**

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

# KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

## HASIL

| | |
|---|---|
| **Mahasiswa** : ROSDIANA | **Pembimbing I** : Muh. Basri, ST., MT. |
| **NIM** : 220280043 | **Pembimbing II** : : Marlina, S.Kom.,M.Kom |
| **Judul Skripsi** : PENGEMBANGAN SISTEM INFORMASI PERPUSTAKAAN MENGGUNAKAN SISTEM BARCODE | |

| ARAHAN PEMBIMBING I | HARI/TGL & PARAF PEMBIMBING | ARAHAN PEMBIMBING II | HARI/TGL & PARAF PEMBIMBING |
|---|---|---|---|
| Konsultasi 1 | | Konsultasi 1<br>- Masukkan Background Pada Tampilan Utama<br>- Input Minimal 20 buku | |
| Konsultasi 2 | | Konsultasi 2<br>- Berikan format Rp pd Aplikasi<br>- Buat Kotak Pesan | |
| Konsultasi 3 | | Konsultasi 3<br>Pemberitahuan apbila peminjaman lebih dari yg ditentukan | |
| Konsultasi 4<br>Acc Ujian<br>yg koreksi<br>Acc Hasil | | Konsultasi 4<br>Lanjutkan ke dokumen | |
| Konsultasi 5 | | Konsultasi 5<br>- Buat Rancangan Database dan Relasinya<br>- Bhs Asing dicetak miring | |

*Lanjut ke halaman sebelah...*

**Perhatian :**

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

*Lanjutan ...*

| ARAHAN PEMBIMBING I | HARI/TGL & PARAF PEMBIMBING | ARAHAN PEMBIMBING II | HARI/TGL & PARAF PEMBIMBING |
|---|---|---|---|
| Konsultasi 6 | | Konsultasi 6<br>- Beri penjelasan pada setiap desainnya. | |
| Konsultasi 7 | | Konsultasi 7<br>Acc Ujian Hasil | |
| Konsultasi 8 | | Konsultasi 8 | |
| Konsultasi 9 | | Konsultasi 9 | |
| Konsultasi 10 | | Konsultasi 10 | |

Parepare, 18 MARET 2025

Mengetahui
Ketua Program Studi

**Marlina, S.Kom.,M.Kom.**
NBM. 1162 680

Mahasiswa

**ROSDIANA**
**NIM. 220280043**

**Perhatian :**

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik