**BAB II**

**Xamarin**

* 1. **Tujuan**

1. Praktikan mampu mengetahui lingkungan pengembangan Xamarin
2. Praktikan memahami struktur file Xamarin
3. Praktikan memahami penggunaan Xamarin Form
4. Praktikan dapat memahami Bagaimana cara untuk menyimpan data secara lokal menggunakan database SQLite.
5. Praktikan memahami penggunaan Application Lifecycle pada Xamarin
6. Praktikan mampu membuat Login pada Xamarin Forms
   1. **Alat dan Bahan**
      1. **PC/Notebook**

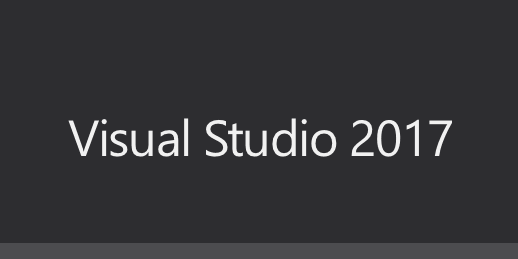
Di praktikum ini Laptop digunakan untuk menginstall Andorid Studio yang akan digunakan untuk membuat program. Laptop minimal memiliki spesifikasi RAM 4GB



Gambar 2.1 PC/Notebook

* + 1. **Visual Studio Community 2017**

Visual Studio program untuk membuat aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup kompiler, SDK, Integrated Development Environment (IDE), dan dokumentasi.



Gambar 2.2 Visual Studio Community 2017

* + 1. **Device Android dan Kabel Data**

Device yang digunakan pada praktikum BAB III ini adalah perangkat yang berjalan pada system operasi Android dengan didukung kabel data USB.



Gambar 2.3 Device Android dan Kabel Data USB

* + 1. **Android SDK**

Android SDK (Software Development Kit) adalah satu set alat pengembangan yang digunakan untuk mengembangkan aplikasi untuk platform Android**.**



Gambar 2.4 Android SDK

**2.3 Dasar Teori**

**2.3.1 Visual Studio**

Microsoft Visual Studio merupakan sebuah [perangkat lunak](https://id.wikipedia.org/wiki/Perangkat_lunak) lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup [kompiler](https://id.wikipedia.org/wiki/Kompiler), [SDK](https://id.wikipedia.org/w/index.php?title=Software_Development_Kit&action=edit&redlink=1), [Integrated Development Environment (IDE)](https://id.wikipedia.org/wiki/Integrated_Development_Environment), dan dokumentasi (umumnya berupa [MSDN Library](https://id.wikipedia.org/w/index.php?title=MSDN_Library&action=edit&redlink=1)). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain [Visual C++](https://id.wikipedia.org/wiki/Visual_C%2B%2B), [Visual C#](https://id.wikipedia.org/w/index.php?title=Visual_C&action=edit&redlink=1), [Visual Basic](https://id.wikipedia.org/wiki/Visual_Basic), [Visual Basic .NET](https://id.wikipedia.org/wiki/Visual_Basic_.NET), [Visual InterDev](https://id.wikipedia.org/w/index.php?title=Visual_InterDev&action=edit&redlink=1), [Visual J++](https://id.wikipedia.org/w/index.php?title=Visual_J%2B%2B&action=edit&redlink=1), [Visual J#](https://id.wikipedia.org/w/index.php?title=Visual_J&action=edit&redlink=1), [Visual FoxPro](https://id.wikipedia.org/wiki/Visual_FoxPro), dan [Visual SourceSafe](https://id.wikipedia.org/w/index.php?title=Visual_SourceSafe&action=edit&redlink=1).

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed code*(dalam bentuk [Microsoft Intermediate Language](https://id.wikipedia.org/wiki/Microsoft_Intermediate_Language) di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi [Silverlight](https://id.wikipedia.org/wiki/Silverlight), aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework).

*(Sumber: Modul 1 Praktikum MDP 2018)*

**2.3.2 Xamarin**

Xamarin adalah platform pengembangan aplikasi mobile dari Microsoft. Dengan Xamarin anda dapat membuat aplikasi mobile cross platform pada perangkat Android, IOS, Windows 10, Windows 8.1, dan Windows Phone. Saat ini terdapat 3 produk Xamarin yang dapat anda pilih yaitu:

* Xamarin for Android
* Xamarin for IOS
* Xamarin Forms

Seperti namanya Xamarin for Android digunakan jika ingin membuat aplikasi native Android. Sedangkan Xamarin for IOS digunakan untuk membuat aplikasi native IOS. Jika menggunakan salah satu atau kedua teknologi tersebut maka dapat membagi komponen yang sudah dibuat pada satu project untuk digunakan di project yang lain. Namun yang dapat dibagi hanya komponen selain UI (User Interface). Xamarin for Android dan Xamarin for IOS masih menggukanan komponen UI yang spesifik dengan platform.

Xamarin Forms adalah teknologi yang lebih baru, kelebihan dari Xamarin Forms adalah dapat menggunakan satu UI yang dapat di gunakan pada platform yang berbeda.

Arsitektur dari platform Xamarin dapat dilihat pada gambar dibawah ini:

Gambar 2.5 Arsitektur dari platform Xamarin

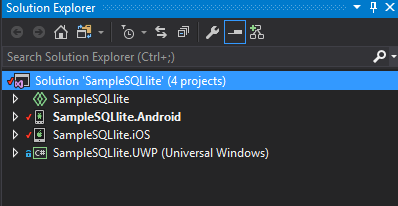
*(Sumber: Modul 1 Praktikum MDP 2018)*

**2.3.3 Xamarin Form**

Xamarin Forms adalah teknologi yang lebih baru, kelebihan dari Xamarin Forms adalah dapat menggunakan satu UI yang dapat di gunakan pada platform yang berbeda. Dengan Xamarin Forms anda cukup mengembangkan satu aplikasi dengan target platform yang berbeda. Xamarin Forms menggunakan format XAML (eXtensible Application Markup Language) untuk membuat tampilan UI.

tampilan solution explorer pada Visual Studio yang berisi 5 project yaitu:

* **Portable**: berisi project utama yang akan dishare secara otomatis ke project yang mempunyai platform spesifik.
* **Droid**: project untuk platform Android.
* **IOS**: project untuk platform IOS.
* **Windows 8.1**: project untuk platform Windows 8.1.
* **Windows Phone 8.1**: project untuk platform Windows Phone 8.1.

**UWP (Windows 10)**– jika anda menggunakan sistem operasi Windows 10 dan menginstal Windows 10 SDK.

Gambar 2.6 Solution explorer pada Visual Studio

1. **Application Lifecycle pada Xamarin**

Xamarin Forms mempunyai application lifecycle yang sederhana. Ada tiga method utama yang digunakan untuk menangani method lifecycle, ketiga proses tersebut adalah:

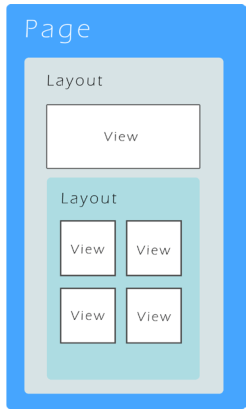
* **OnStart**: method ini akan dipanggil ketika aplikasi pertama kali dijalankan.
* **OnSleep**: method ini akan dipanggil setiap kali app berpindah ke mode background
* **OnResume**: method ini akan dipanggil ketika app masuk ke mode resume setelah sebelumnya masuk ke mode background.

Ketika pengguna memilih tombol Back atau Home pada ponsel, maka aplikasi yang sedang aktif pada saat itu akan masuk ke mode background. Ketika pengguna memilih kembali aplikasi tersebut, maka app tersebut di resume dan app tersebut dibawa kembali ke mode foreground. Saat ketika aplikasi dijalankan pertama kali, saat app masuk ke mode background dan masuk ke mode foreground lagi, sampai app tersebut ditutup (terminate).

1. **Xamarin Form UI (User Interface)**

Ada tiga komponen utama yang membentuk Xamarin Forms UI yaitu:

* **Page**: Halaman yang digunakan untuk menampilkan view yang disusun menggunakan layout.
* **Views**: komponen kontrol yang ada pada Xamarin Forms seperti Entry, Button, dll.
* **Layout**: komponen yang digunakan untuk mengatur posisi views pada halaman.



Gambar 2.7 Xamarin Form UI

1. **Page**

Class Page adalah kontainer utama untuk setiap tampilan pada halaman aplikasi di Xamarin Forms. Class ini diturunkan dari class **Xamarin.Forms.VisualElement**. Class Page adalah baseclass untuk membuat dari semua class UI pada Xamarin Forms. Berikut adalah beberapa contoh Page yang umum digunakan:

* **ContentPage**
* **MasterDetailPage**
* **NavigationPage**
* **TabbedPage**
* **CarouselPage**

1. **View**

View adalah kontrol interaktif yang ada pada Page. Berikut ini adalah jenis-jenis view yang ada pada Xamarin Forms.

* **Basic – fundamental views**
  + **Label**
  + **Image**
  + **Button**
  + **BoxView**
* **List – scrollabe dan selectable list**
  + **ListView**
* **Text Entry – entry input pengguna menggunakan keyboard**
  + **Entry**
  + **Editor**
* **Selection – pilihan pengguna yang lebih dari satu.**
  + **Picker**
  + **DatePicker**
  + **TimePicker**
  + **Stepper**
  + **Slider**
  + **Switch**
* **User Feedback – notifikasi pengguna**
  + **Activity**

1. **Layout**

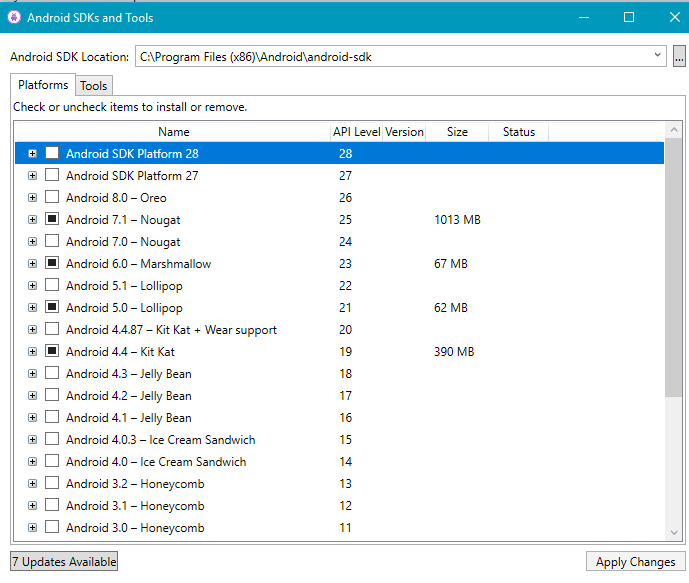
Layout adalah wadah yang digunakan untuk pengaturan posisi kontrol (view, atau layout lain). Ada beberapa macam layout yang didukung oleh Xamarin Form yaitu:

* **StackLayout**: mengatur kontrol secara horisontal atau vertikal.
* **AbsoluteLayout**: pengaturan posisi berdasarkan letak yang pasti.
* **RelativeLayout**: pengaturan posisi kontrol berdasarkan kontrol yang lain.
* **Grid**: membuat layout yang terdiri dari kolom dan baris seperti tabel.

*(Sumber: Modul 1 Praktikum MDP 2018)*

**2.3.4 SDK Pada Xamarin**

Android menggunakan beberapa pengaturan level API Android untuk menentukan kompatibilitas aplikasi yang dibuild di berbagai versi Android. Tergantung pada level Android API yang ditargetkan, sehingga diperlukan mengunduh dan memasang komponen Android SDK tambahan. Selain itu, mungkin perlu menginstal tool opsional dan gambar emulator yang disediakan di Android SDK. Untuk melakukannya, dapat dengan menggunakan Android SDK Manager. Pada Visual Studio 2017 Android SDK Manager dapat disetting dengan mengklik Tools> Android> Android SDK Manager.



Gambar 2.8 SDK pada Xamarin

*(Sumber: Modul 1 Praktikum MDP 2018)*

**2.3.5 iOS**

iOS (sebelumnya iPhone OS) adalah sistem operasi perangkat bergerak yang dikembangkan dan didistribusikan oleh Apple Inc. Sistem operasi ini pertama diluncurkan tahun 2007 untuk iPhone dan iPod Touch, dan telah dikembangkan untuk mendukung perangkat Apple lainnya seperti iPad dan Apple TV.

Tidak seperti IDE lain pada Xamarin.Forms aplikasi yang dibuat dapat kompatibel pada 3 platform sekaligus termasuk salah satunya iOS.



Gambar 2.9 Sistem Operasi iOS

*(Sumber: Modul 1 Praktikum MDP 2018)*

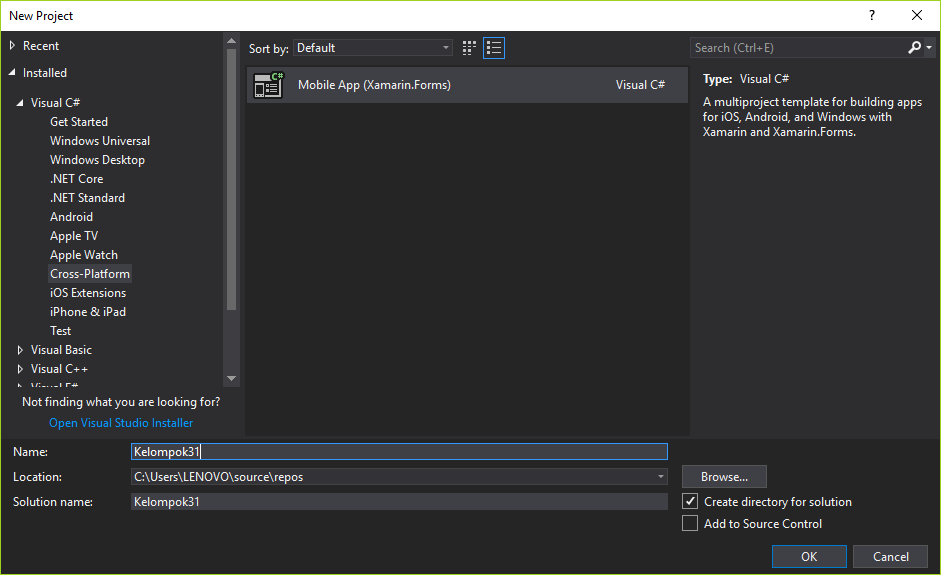
**2.3.6 SQLite**

SQLite merupakan sebuah [sistem manajemen basisdata relasional](https://id.wikipedia.org/wiki/Sistem_manajemen_basisdata_relasional) yang bersifat [ACID](https://id.wikipedia.org/w/index.php?title=ACID&action=edit&redlink=1)-compliant dan memiliki ukuran pustaka kode yang relatif kecil, ditulis dalam [bahasa C](https://id.wikipedia.org/wiki/Bahasa_pemrograman_C). SQLite merupakan proyek yang bersifat [public domain](https://id.wikipedia.org/wiki/Public_domain) yang dikerjakan oleh [D. Richard Hipp](https://id.wikipedia.org/w/index.php?title=D._Richard_Hipp&action=edit&redlink=1).

Tidak seperti pada paradigma [client-server](https://id.wikipedia.org/w/index.php?title=Client-server&action=edit&redlink=1) umumnya, Inti SQLite bukanlah sebuah sistem yang mandiri yang berkomunikasi dengan sebuah program, melainkan sebagai bagian integral dari sebuah program secara keseluruhan. Sehingga protokol komunikasi utama yang digunakan adalah melalui pemanggilan [API](https://id.wikipedia.org/wiki/API) secara langsung melalui bahasa pemrograman. Mekanisme seperti ini tentunya membawa keuntungan karena dapat mereduksi *overhead*, *latency times*, dan secara keseluruhan lebih sederhana. Seluruh elemen basisdata (definisi data, tabel, indeks, dan data) disimpan sebagai sebuah file. Kesederhanaan dari sisi disain tersebut bisa diraih dengan cara mengunci keseluruhan file basis data pada saat sebuah transaksi dimulai.

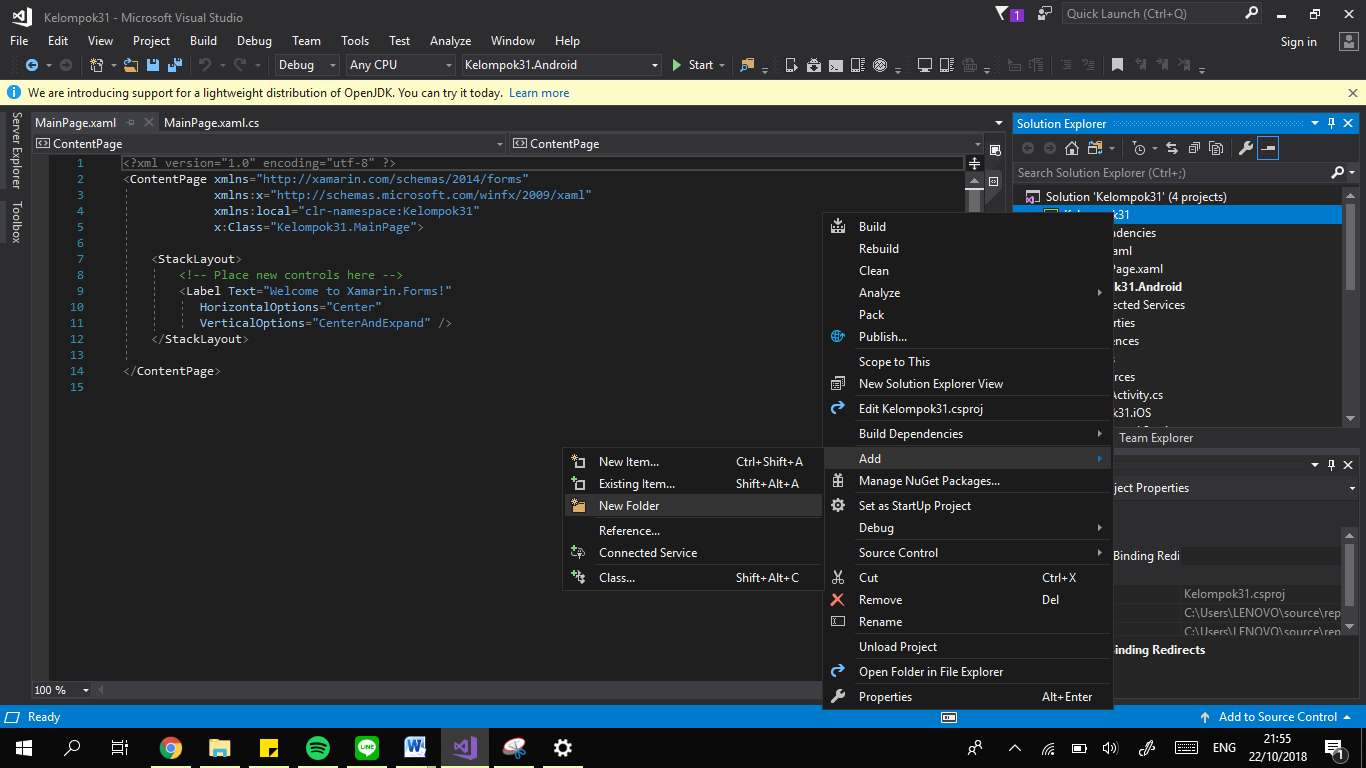
*(Sumber: Modul 1 Praktikum MDP 2018)*

* 1. **Langkah Kerja**
     1. **Membuat Login**
  2. Membuat project baru dengan nama Kelompok31. Pilih Cross-Platform App (Xamarin.Forms).



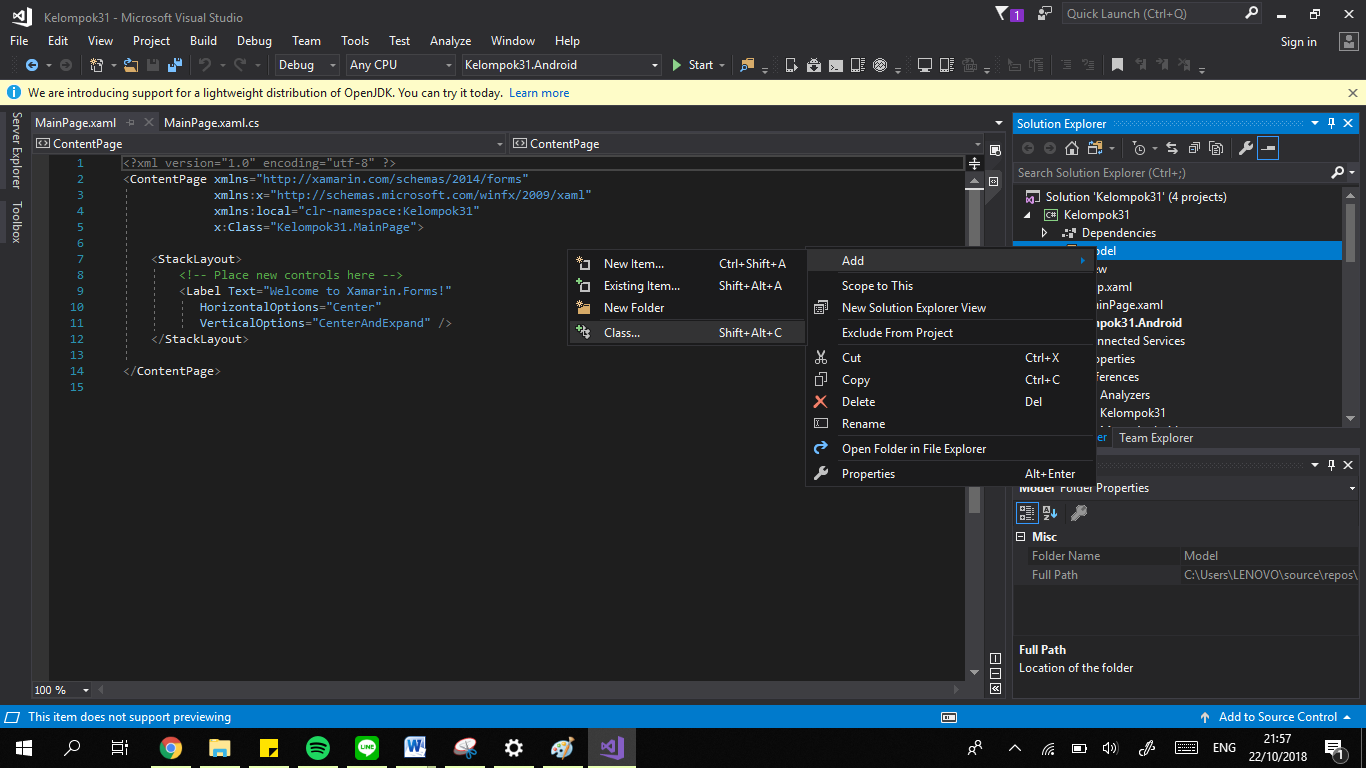
Gambar 2.10 Membuat Xamarin Forms

* 1. Membuat folder View dan Model pada project Xamarin.Forms Kelompok31. Klik kanan pada project Kelompok31 > Add > New Folder.

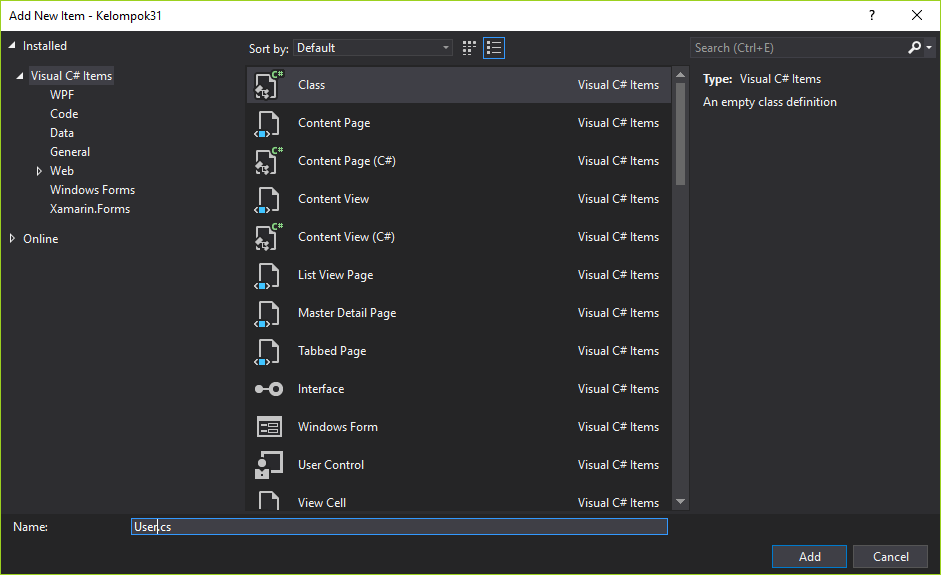


Gambar 2.11 Membuat Folder baru

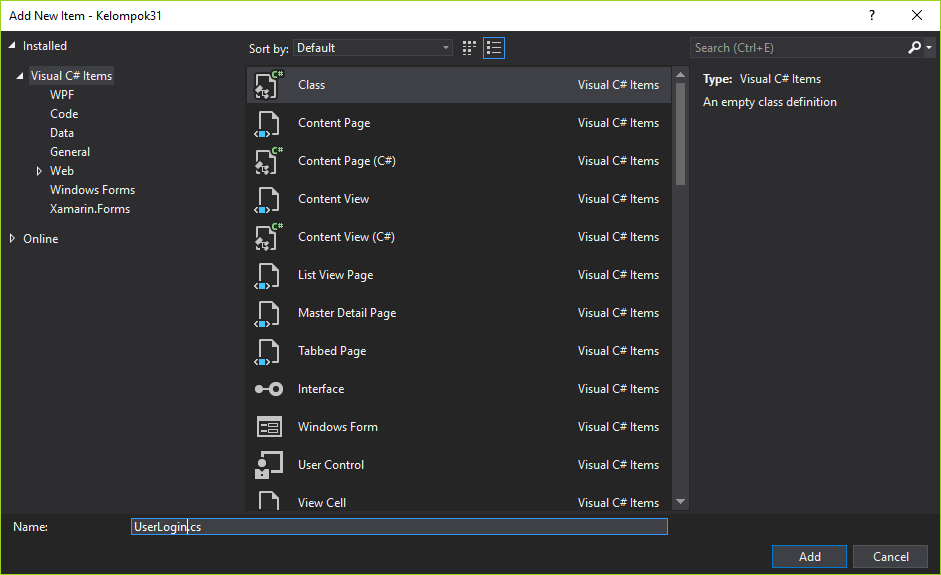
* 1. Membuat Class User.cs dan UserLogin.cs pada direktori atau folder model yang sudah dibuat sebelumnya. Klik kanan pada folder Model > Add > Class



Gambar 2.12 Menambahakan File Class

****

Gambar 2.13 Memilih tipe file class dan Penamaan class User.cs



Gambar 2.14 Memilih tipe file class dan Penamaan class UserLogin.cs

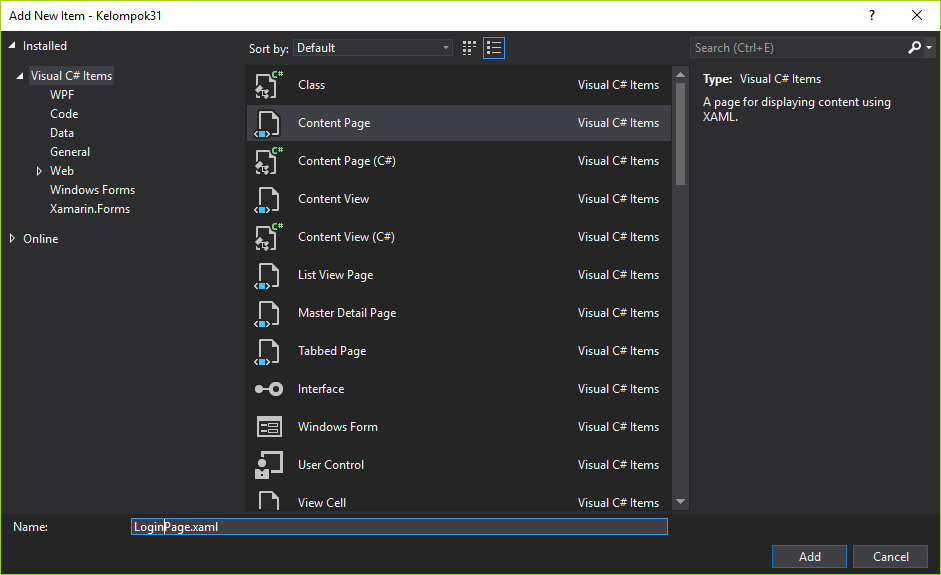
* 1. Masukkan *source code* di bawah ini ke dalam file User.cs. Pada file ini terdapat beberapa parameter yang digunakan untuk melakukan SignUp.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Text;  namespace Kelompok31.Model  {  public class User  {  public string Username { get; set; }  public string Password { get; set; }  public string Email { get; set; }  }  } |

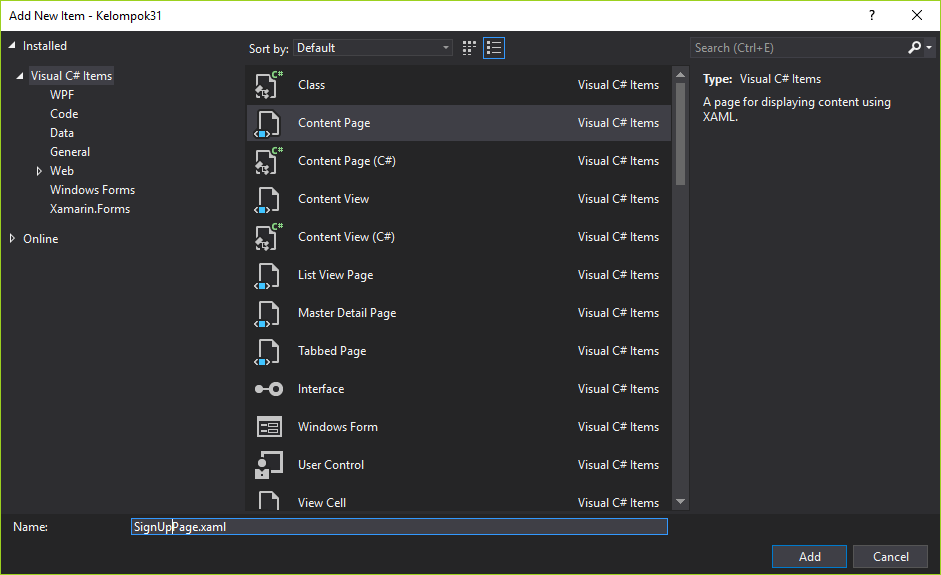
* 1. Masukkan source code berikut ini ke dalam file UserLogin.cs. Pada file ini terdapat dua variabel yang digunakan sebagai user dan password ketika akan melakukan login ke dalam aplikasi.

|  |
| --- |
| namespace Kelompok31  {  public static class UserLogin  {  public static string Username = "Kelompok31";  public static string Password = "password";  }  } |

* 1. Membuat Content Page, masing – masing file diberi nama LoginPage.xaml, SignUpPage.xaml. Pada file Content Page terdapat dua jenis file sekaligus yaitu file dengan format .xaml dan .cs. Klik kanan pada project Kelompok31 > Add > New item > Content Page



Gambar 2.15 Menambahkan Contant Page LoginPage.xaml



Gambar 2.16 Menambahkan Contant Page SingUpPage.xaml

* 1. Masukkan *source code* dibawah ini ke dalam file LoginPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="Kelompok31.LoginPage" Title="Login">  <ContentPage.ToolbarItems>  <ToolbarItem Text="Sign Up" Clicked="OnSignUpButtonClicked" />  </ContentPage.ToolbarItems>  <ContentPage.Content>  <StackLayout VerticalOptions="StartAndExpand">  <Label Text="Username" />  <Entry x:Name="usernameEntry" Placeholder="Masukkan Username" />  <Label Text="Password" />  <Entry x:Name="passwordEntry" IsPassword="true" Placeholder="Masukkan Password"/>  <Button Text="Login" Clicked="OnLoginButtonClicked" />  <Label x:Name="messageLabel" />  </StackLayout>  </ContentPage.Content>  </ContentPage> |

* 1. Masukkan *source code* dibawah ini ke dalam file LoginPage.xaml.cs

|  |
| --- |
| using Kelompok31.Model;  using Kelompok31.View;  using System;  using Xamarin.Forms;  namespace Kelompok31  {  public partial class LoginPage : ContentPage  {  public LoginPage()  {  InitializeComponent();  }  async void OnSignUpButtonClicked(object sender, EventArgs e)  {  await Navigation.PushAsync(new SignUpPage());  }  async void OnLoginButtonClicked(object sender, EventArgs e)  {  var user = new User  {  Username = usernameEntry.Text,  Password = passwordEntry.Text  };  var isValid = AreCredentialsCorrect(user);  if (isValid)  {  App.IsUserLoggedIn = true;  Navigation.InsertPageBefore(new HalamanUtama(), this);  await Navigation.PopAsync();  }  else  {  messageLabel.Text = "Login failed";  passwordEntry.Text = string.Empty;  }  }  bool AreCredentialsCorrect(User user)  {  return user.Username == UserLogin.Username && user.Password == UserLogin.Password;  }  }  } |

* 1. Masukkan *source code* berikut kedalam file SignUpPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="Kelompok31.SignUpPage"  Title="Sign Up">  <ContentPage.Content>  <StackLayout VerticalOptions="StartAndExpand">  <Label Text="Username" />  <Entry x:Name="usernameEntry" Placeholder="username" />  <Label Text="Password" />  <Entry x:Name="passwordEntry" IsPassword="true" />  <Label Text="Email address" />  <Entry x:Name="emailEntry" />  <Button Text="Sign Up" Clicked="OnSignUpButtonClicked" />  <Label x:Name="messageLabel" />  </StackLayout>  </ContentPage.Content>  </ContentPage> |

* 1. Masukkan *source code* berikut ke dalam file SignUpPage.xaml.cs

|  |
| --- |
| using Kelompok31.Model;  using System;  using System.Linq;  using Xamarin.Forms;  namespace Kelompok31  {  public partial class SignUpPage : ContentPage  {  public SignUpPage()  {  InitializeComponent();  }  async void OnSignUpButtonClicked(object sender, EventArgs e)  {  var user = new User()  {  Username = usernameEntry.Text,  Password = passwordEntry.Text,  Email = emailEntry.Text  };  // Sign up logic goes here  var signUpSucceeded = AreDetailsValid(user);  if (signUpSucceeded)  {  var rootPage = Navigation.NavigationStack.FirstOrDefault();  if (rootPage != null)  {  App.IsUserLoggedIn = true;  Navigation.InsertPageBefore(new MainPage(), Navigation.NavigationStack.First());  await Navigation.PopToRootAsync();  }  }  else  {  messageLabel.Text = "Sign up failed";  }  }  bool AreDetailsValid(User user)  {  return (!string.IsNullOrWhiteSpace(user.Username) && !string.IsNullOrWhiteSpace(user.Password) && !string.IsNullOrWhiteSpace(user.Email) && user.Email.Contains("@"));  }  }  } |

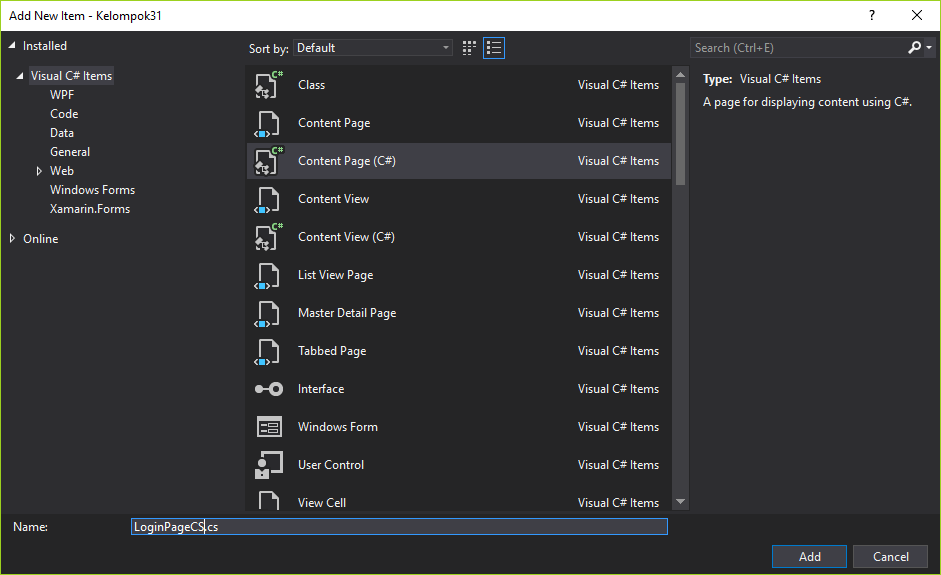
* 1. Masukkan *source code* berikut ke dalam file MainPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="Kelompok31.MainPage" Title="Main Page">  <ContentPage.ToolbarItems>  <ToolbarItem Text="Logout" Clicked="OnLogoutButtonClicked" />  </ContentPage.ToolbarItems>  <ContentPage.Content>  <StackLayout>  <Label Text="Main app content goes here" HorizontalOptions="Center" VerticalOptions="CenterAndExpand" />  </StackLayout>  </ContentPage.Content>  </ContentPage> |

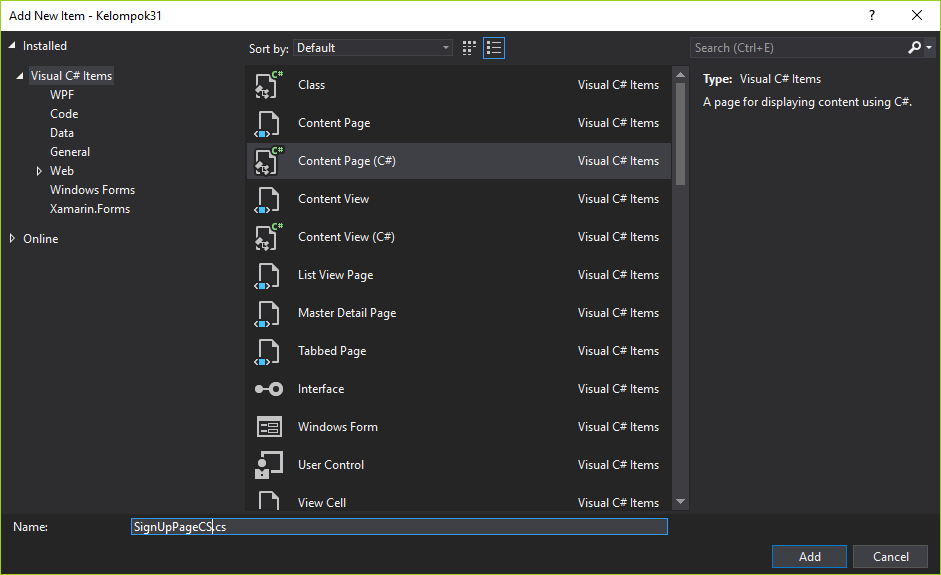
* 1. Masukkan *source code* berikut ini ke dalam file MainPage.xaml.cs

|  |
| --- |
| using Kelompok31;  using System;  using Xamarin.Forms;  namespace Kelompok31  {  public partial class MainPage : ContentPage  {  public MainPage()  {  InitializeComponent();  }  async void OnLogoutButtonClicked(object sender, EventArgs e)  {  App.IsUserLoggedIn = false;  Navigation.InsertPageBefore(new LoginPage(), this);  await Navigation.PopAsync();  }  }  } |

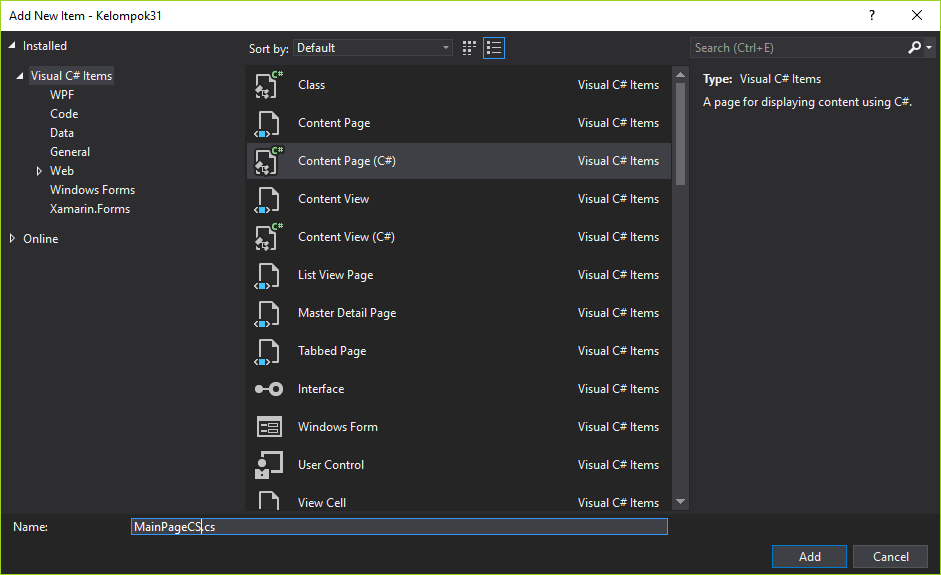
* 1. Kemudian langkah selanjutnya yaitu membuat Content Page (C#), masing – masing file dibuat dan diberi nama dengan LoginPageCS.cs, MainPageCS.cs dan SignUpPageCS.cs. Klik kanan pada project Kelompok31 > Add > New item > Content Page (C#)



Gambar 2.17 Membuat Contant Page LoginPageCS.cs



Gambar 2.18 Membuat Contant Page SignUpPageCS.cs



Gambar 2.19 Membuat Contant Page MainPageCS.cs

* 1. Masukkan *source code* dibawah ini kedalam file LoginPageCS.cs

|  |
| --- |
| using Kelompok31.Model;  using System;  using Xamarin.Forms;  namespace Kelompok31  {  public class LoginPageCS : ContentPage  {  Entry usernameEntry, passwordEntry;  Label messageLabel;  public LoginPageCS()  {  var toolbarItem = new ToolbarItem  {  Text = "Sign Up"  };  toolbarItem.Clicked += OnSignUpButtonClicked;  ToolbarItems.Add(toolbarItem);  messageLabel = new Label();  usernameEntry = new Entry  {  Placeholder = "username"  };  passwordEntry = new Entry  {  IsPassword = true  };  var loginButton = new Button  {  Text = "Login"  };  loginButton.Clicked += OnLoginButtonClicked;  Title = "Login";  Content = new StackLayout  {  VerticalOptions = LayoutOptions.StartAndExpand,  Children = {  new Label { Text = "Username" },  usernameEntry,  new Label { Text = "Password" },  passwordEntry,  loginButton,  messageLabel  }  };  }  async void OnSignUpButtonClicked(object sender, EventArgs e)  {  await Navigation.PushAsync(new SignUpPageCS());  }  async void OnLoginButtonClicked(object sender, EventArgs e)  {  var user = new User  {  Username = usernameEntry.Text,  Password = passwordEntry.Text  };  var isValid = AreCredentialsCorrect(user);  if (isValid)  {  App.IsUserLoggedIn = true;  Navigation.InsertPageBefore(new MainPageCS(), this);  await Navigation.PopAsync();  }  else  {  messageLabel.Text = "Login failed";  passwordEntry.Text = string.Empty;  }  }  bool AreCredentialsCorrect(User user)  {  return user.Username == UserLogin.Username && user.Password == UserLogin.Password;  }  }  } |

* 1. Masukkan *source code* dibawah ini kedalam file MainPageCS.cs

|  |
| --- |
| using System;  using Xamarin.Forms;    namespace Kelompok31  {  public class MainPageCS : ContentPage  {  public MainPageCS()  {  var toolbarItem = new ToolbarItem  {  Text = "Logout"  };  toolbarItem.Clicked += OnLogoutButtonClicked;  ToolbarItems.Add(toolbarItem);  Title = "Main Page";  Content = new StackLayout  {  Children = {  new Label {  Text = "Main app content goes here",  HorizontalOptions = LayoutOptions.Center,  VerticalOptions = LayoutOptions.CenterAndExpand  }  }  };  }  async void OnLogoutButtonClicked(object sender, EventArgs e)  {  App.IsUserLoggedIn = false;  Navigation.InsertPageBefore(new LoginPageCS(), this);  await Navigation.PopAsync();  }  }  } |

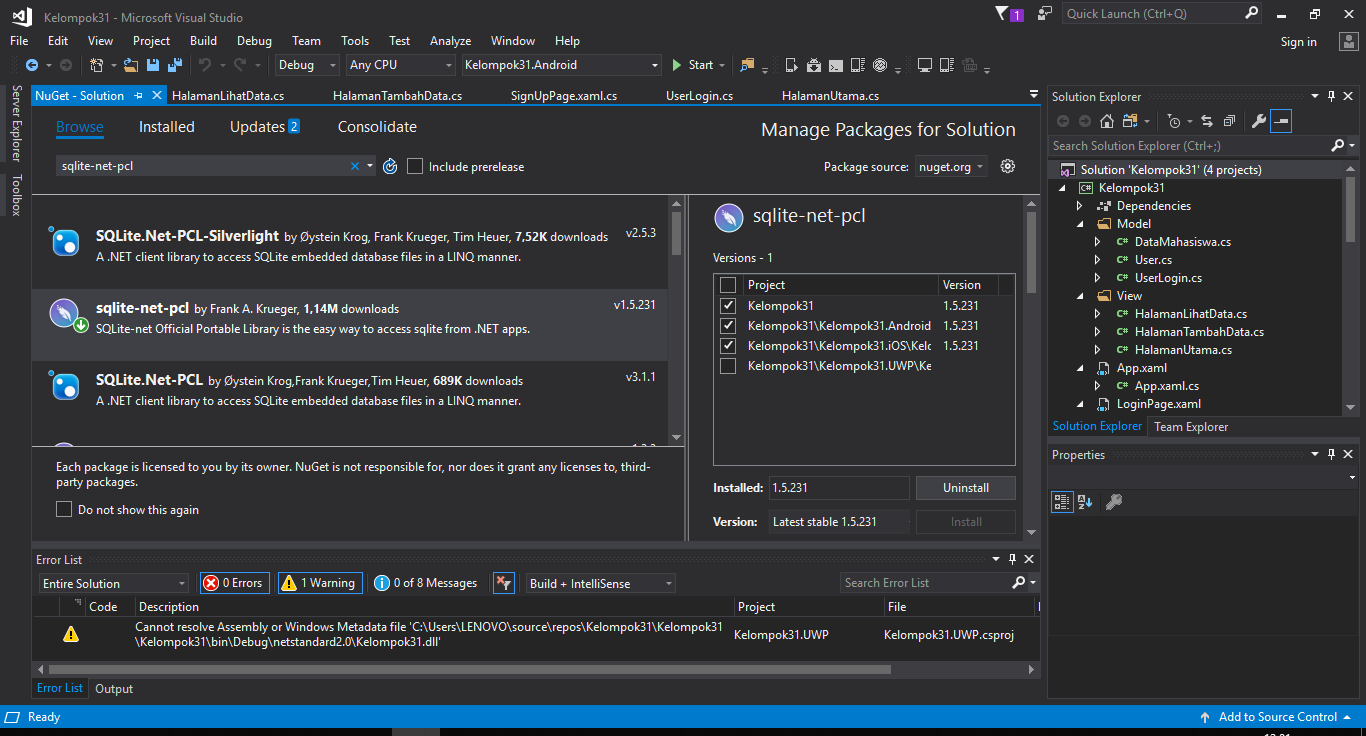
* 1. Masukkan *source code* berikut kedalam file SignUpPageCS.cs

|  |
| --- |
| using Kelompok31;  using Kelompok31.Model;  using System;  using System.Linq;  using Xamarin.Forms;  namespace Kelompok31  {  public class SignUpPageCS : ContentPage  {  Entry usernameEntry, passwordEntry, emailEntry;  Label messageLabel;  public SignUpPageCS()  {  messageLabel = new Label();  usernameEntry = new Entry  {  Placeholder = "username"  };  passwordEntry = new Entry  {  IsPassword = true  };  emailEntry = new Entry();  var signUpButton = new Button  {  Text = "Sign Up"  };  signUpButton.Clicked += OnSignUpButtonClicked;  Title = "Sign Up";  Content = new StackLayout  {  VerticalOptions = LayoutOptions.StartAndExpand,  Children = {  new Label { Text = "Username" },  usernameEntry,  new Label { Text = "Password" },  passwordEntry,  new Label { Text = "Email address" },  emailEntry,  signUpButton,  messageLabel  }  };  }  async void OnSignUpButtonClicked(object sender, EventArgs e)  {  var user = new User()  {  Username = usernameEntry.Text,  Password = passwordEntry.Text,  Email = emailEntry.Text  };  // Sign up logic goes here  var signUpSucceeded = AreDetailsValid(user);  if (signUpSucceeded)  {  var rootPage = Navigation.NavigationStack.FirstOrDefault();  if (rootPage != null)  {  App.IsUserLoggedIn = true;  Navigation.InsertPageBefore(new MainPageCS(), Navigation.NavigationStack.First());  await Navigation.PopToRootAsync();  }  }  else  {  messageLabel.Text = "Sign up failed";  }  }  bool AreDetailsValid(User user)  {  return (!string.IsNullOrWhiteSpace(user.Username) && !string.IsNullOrWhiteSpace(user.Password) && !string.IsNullOrWhiteSpace(user.Email) && user.Email.Contains("@"));  }  }  } |

* + 1. **Percobaan SQLite**

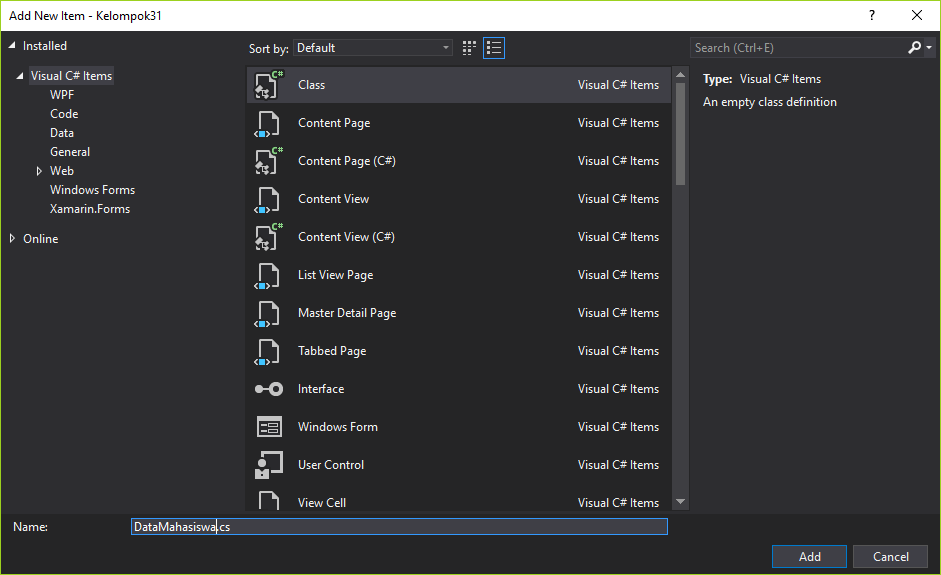
1. Klik kanan pada project Solution Kelompok31 kemudian pilih menu Manage NuGet Packages for Solution.
2. Pada tab Browse terdapat kolom pencarian NuGet Packages yang akan dicari. Kemudian ketik “sqlite-net-pcl” setelah itu terdapat checkbox pada project yang akan diinstall NuGet Packages yang sudah dipilih.

Apabila NuGet Packages sudah terinstall maka button yang akan muncul yaitu button Uninstall.



Gambar 2.20 Petunjuk yang harus dilakukan

1. Membuat Class baru pada folder Model yang diberi nama DataMahasiswa.cs. Pada File ini terdapat parameter data apa saja yang akan kita gunakan pada aplikasi. Klik kanan pada folder Model > Add > Class

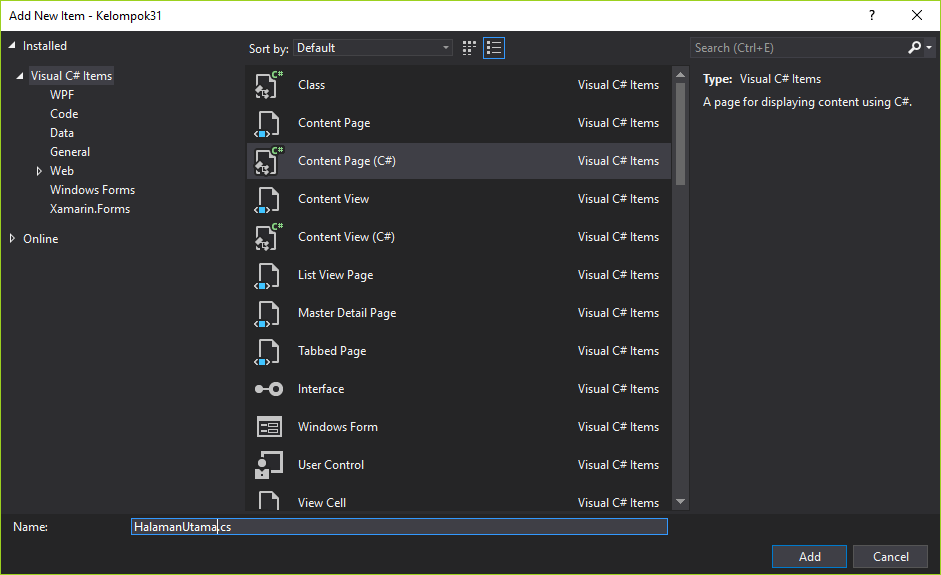


Gambar 2.21 Membuat file class baru Data mahasiswa

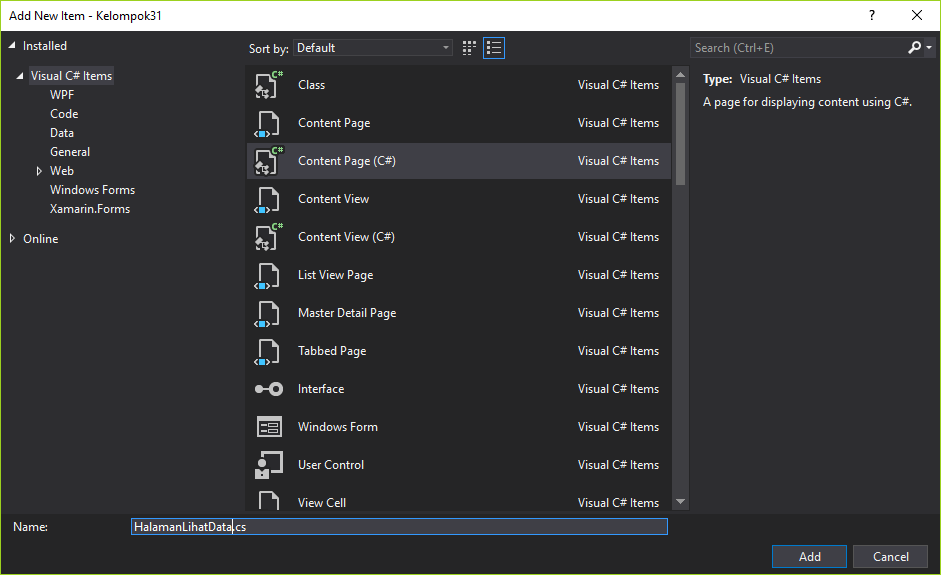
1. Ketik *source code* berikut ke dalam DataMahasiswa.cs

|  |
| --- |
| using SQLite;  using System;  using System.Collections.Generic;  using System.Text;  namespace Kelompok31.Mode1  {  public class DataMahasiswa  {  [PrimaryKey]  public int Id {get; set; }  public string Nama {get; set; }  public string Jurusan {get; set; }  public override string ToString()  {  return “Nama : “+this.Nama+ “ ==> Jurusan : “+ this.Jurusan;  }  }  } |

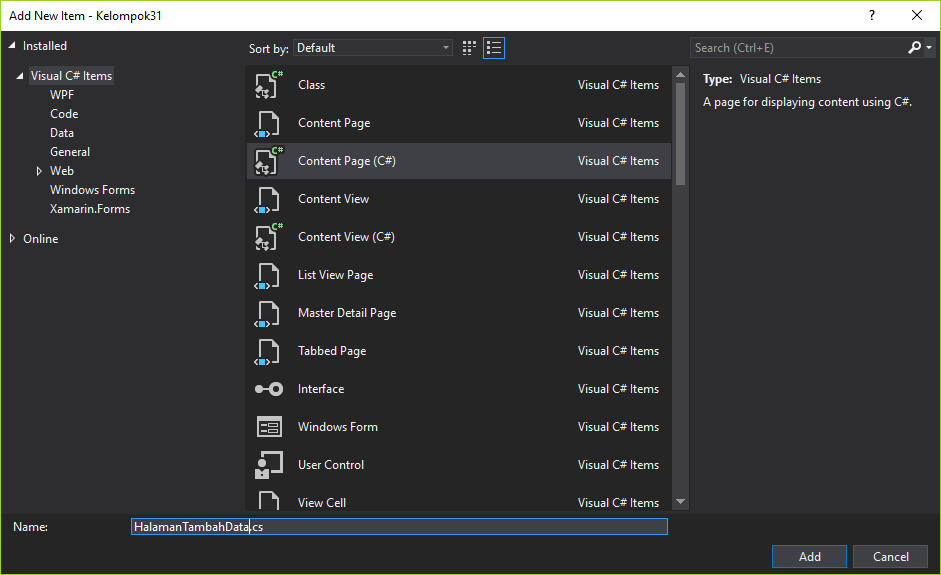
1. Membuat file Content Page (C#) pada folder View, masing – masing file diberi nama HalamanUtama.cs, HalamanTambahData.cs dan HalamanLihatData.cs. Klik kanan pada folder View> Add > New item > Content Page (C#)



Gambar 2.22 Membuat file HalamanUtama.cs *content Page*



Gambar 2.23 Membuat file HalamanLihatData.cs *content Page*



Gambar 2.24 Membuat file HalamanTambahData.cs *content Page*

1. Ketik *source code* berikut ke dalam HalamanUtama.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Xamarin.Forms;  namespace Kelompok31  {  public class HalamanUtama : ContentPage  {  public HalamanUtama ()  {  this.Title = “Data Mahasiswa”;  StackLayout stackLayout = new StackLayout();  Button button = new Button();  button.Text = “Tambah Data”;  button.Clicked += Button\_Tambah\_Clicked;  stackLayout.Children.Add(button);  button = new Button();  button.Text = “Lihat Data”;  button.Clicked += Button\_Lihat\_Clicked;  stackLayout.Children.Add(button);  Content = stackLayout;  }  private async void Button\_Tambah\_Clicked(object sender, EventArgs e)  {  await Navigation.PushAsync(new HalamanTambahData());  }  private async void Button\_Lihat\_Clicked(object sender, EventArgs e)  {  await Navigation.PushAsync(new HalamanLihatData());  }  }  } |

1. Ketik *source code* berikut ke dalam HalamanTambahData.cs

|  |
| --- |
| using Kelompok31.Mode1;  using SQLite;  using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Text;  using Xamarin.Forms;  namespace Kelompok31.View  {  public class HalamanTambahData : ContentPage  {  private Entry \_nama;  private Entry \_jurusan;  private Entry \_simpan;  string \_dbPath = Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), “myDB.db4”);  public HalamanTambahData ()  {  this.Title = “Tambah Data Mahasiswa”;  StackLayout stackLayout = new StackLayout();  \_nama = new Entry();  \_nama.Keyboard = Keyboard.Text;  \_nama.Placeholder = “Nama Mahasiswa”;  stackLayout.Children.Add(\_nama);  \_jurusan = new Entry();  \_jurusan.Keyboard = Keyboard.Text;  \_jurusan.Placeholder = “Jurusan”;  stackLayout.Children.Add(\_jurusan);  \_simpan = new Button();  \_simpan.Text = “Tambah”;  \_simpan.Clicked.Add(\_simpan);  Content = stackLayout;  }  private async void \_simpan\_Clicked(object sender, EventArgs e)  {  var db = new SQLiteConnection(\_dbPath);  db.CreateTable<DataMahasiswa>();  var maxPk = db.Table<DataMahasiswa>().OrderByDescending(c => c.Id).FirstOrDefault();  DataMahasiswa dbm = new DataMahasiswa()  {  Id = (maxPk == null ? 1 : maxPk.Id + 1),  Nama = \_nama.Text,  Jurusan = \_jurusan.Text  };  db.Insert(dbm);  await DisplayAlert(null, “Data “ + dbm.Nama + “ Berhasil Disimpan”, “Ok”);  await Navigation.PopAsync();  }  }  } |

1. Ketik *source code* berikut ke HalamanLihatData.cs

|  |
| --- |
| using Kelompok31.Mode1;  using SQLite;  using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Text;  using Xamarin.Forms;  namespace Kelompok31.View  {  public class HalamanLihatData : ContentPage  {  private ListView \_listView;  string \_dbPath = Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), “myDB.db4”);  public HalamanLihatData ()  {  this.Title = “Data Mahasiswa”;  var db = new SQLiteConnection(\_dbPath);  StackLayout stackLayout = new StackLayout();  \_listView = new ListView();  \_listView.ItemSource = db.Table<DataMahasiswa>().OrderBy(x => x.Nama).ToList();  stackLayout.Children.Add(\_listView);  Content = stackLayout;  }  }  } |

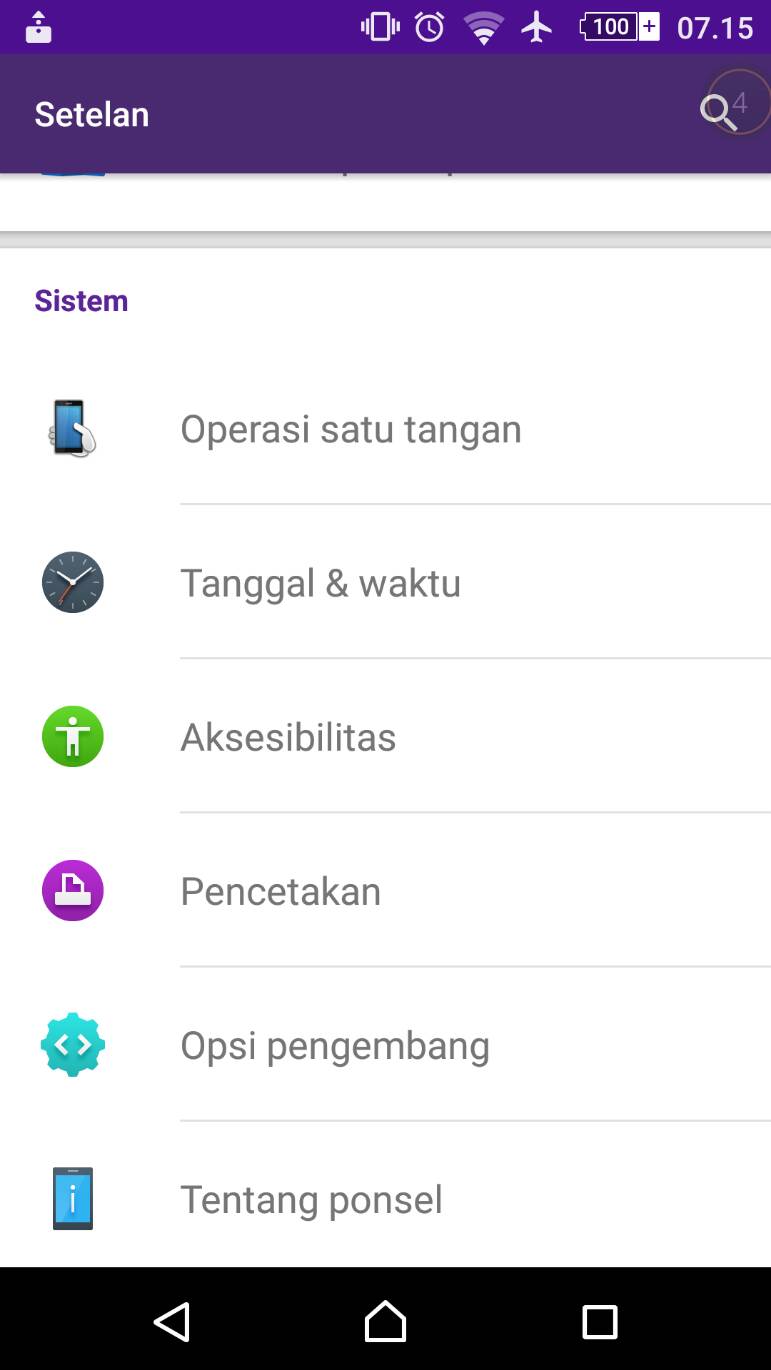
1. Pada file HalamanUtama.cs, HalamanTambahData.cs dan HalamanLihatData.cs diatas terdapat variabel \_dbpath yang di dalamnya merupakan database lokal dengan nama **myDB.db4**. File database tersebut akan disimpan kedalam folder pada device android anda yaitu pada folder **/data/data/[your.package.name]/files**.
2. Buka kembali *Manage Pakages for solutions* dengan cara Clik kanan pada Solutions “Kelompok31”.
3. Klick *Browse* dan ketikan “Xamarin Forms” Pilih kemudian centang pada projek kita lalu pililah yang versi 2.5.0.122203 .
4. Kemudian kita cari file App.xaml pada *Project* kelompok31 Rubah *source code* seperti dibawah pada App.xml.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <Application xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="Kelompok31.App">  <Application.Resources>  </Application.Resources>  </Application> |

1. Kemudian rubah juga *file* app.xaml.cs dengan source code seperti dibawah

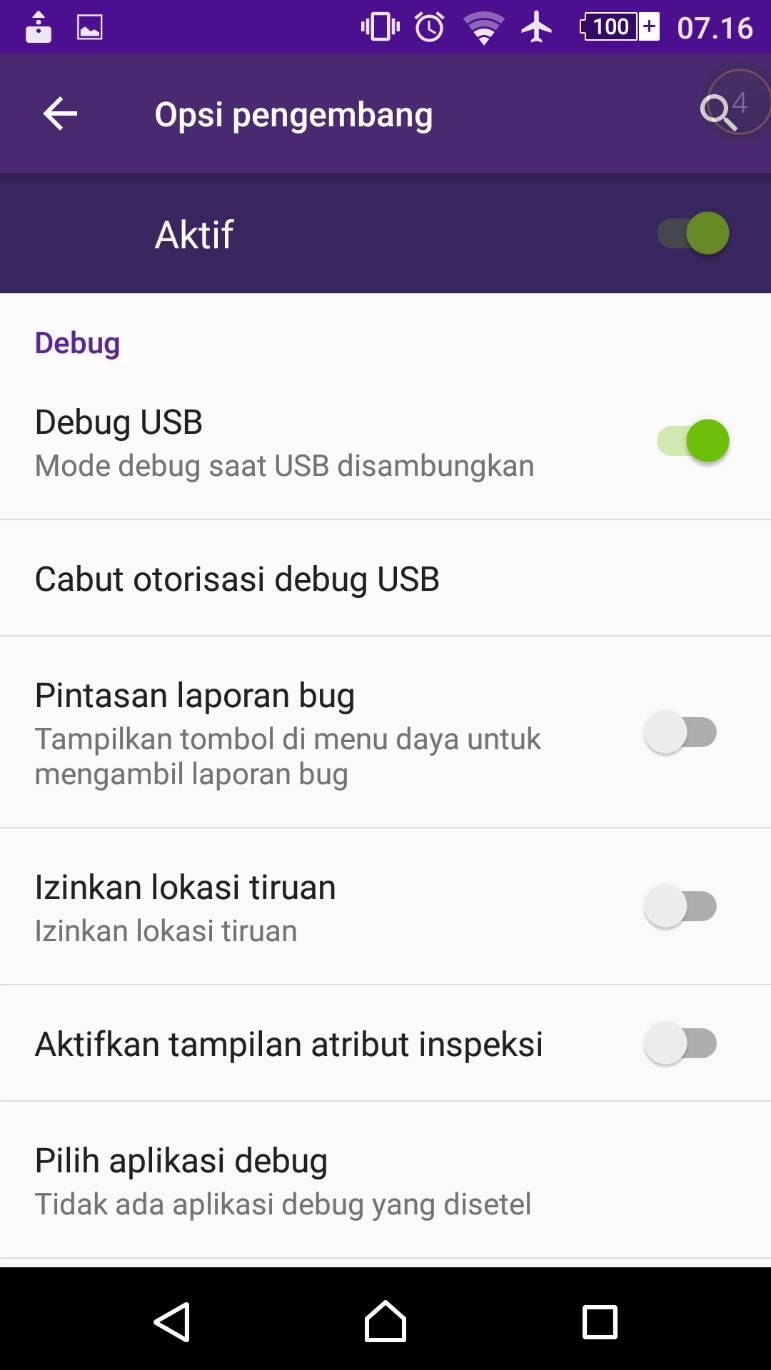
|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Xamarin.Forms;  using Xamarin.Forms.Xaml;  namespace Kelompok31  {  public partial class App : Application  {  public static bool IsUserLoggedIn { get; internal set; }  public App()  {  //InitializeComponent();  if (!IsUserLoggedIn)  {  MainPage = new NavigationPage(new LoginPage());  }  else  {  MainPage = new NavigationPage(new HalamanUtama());  }    }  protected override void OnStart()  {  // Handle when your app starts  }  protected override void OnSleep()  {  // Handle when your app sleeps  }  protected override void OnResume()  {  // Handle when your app resumes  }  }  } |

1. Sebelum melakukan Build Project dan Deploy Project pastikan smartphone sudah terdapat menu Opsi Pengembangan.



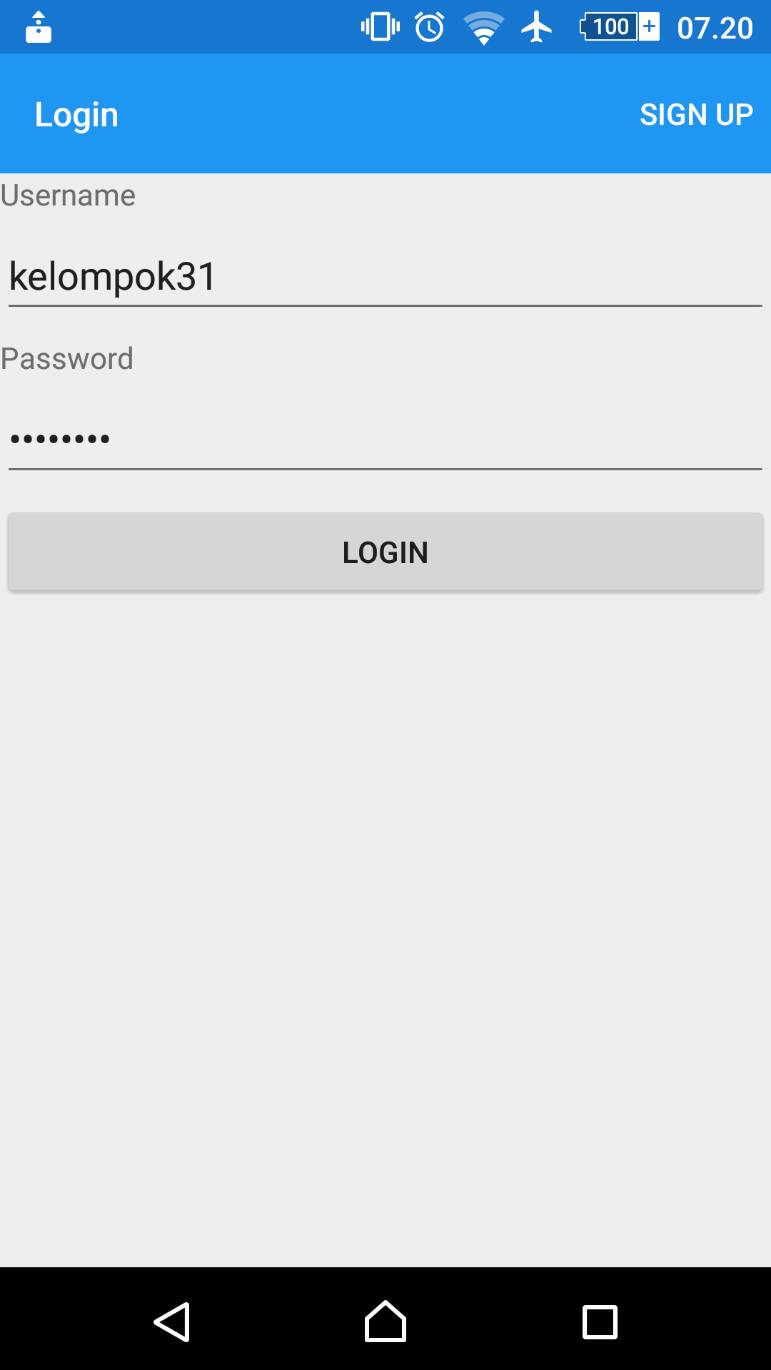
Gambar 2.25 Menu Settings Android

1. Pada menu Developer options checklist pada checkbox USB debugging



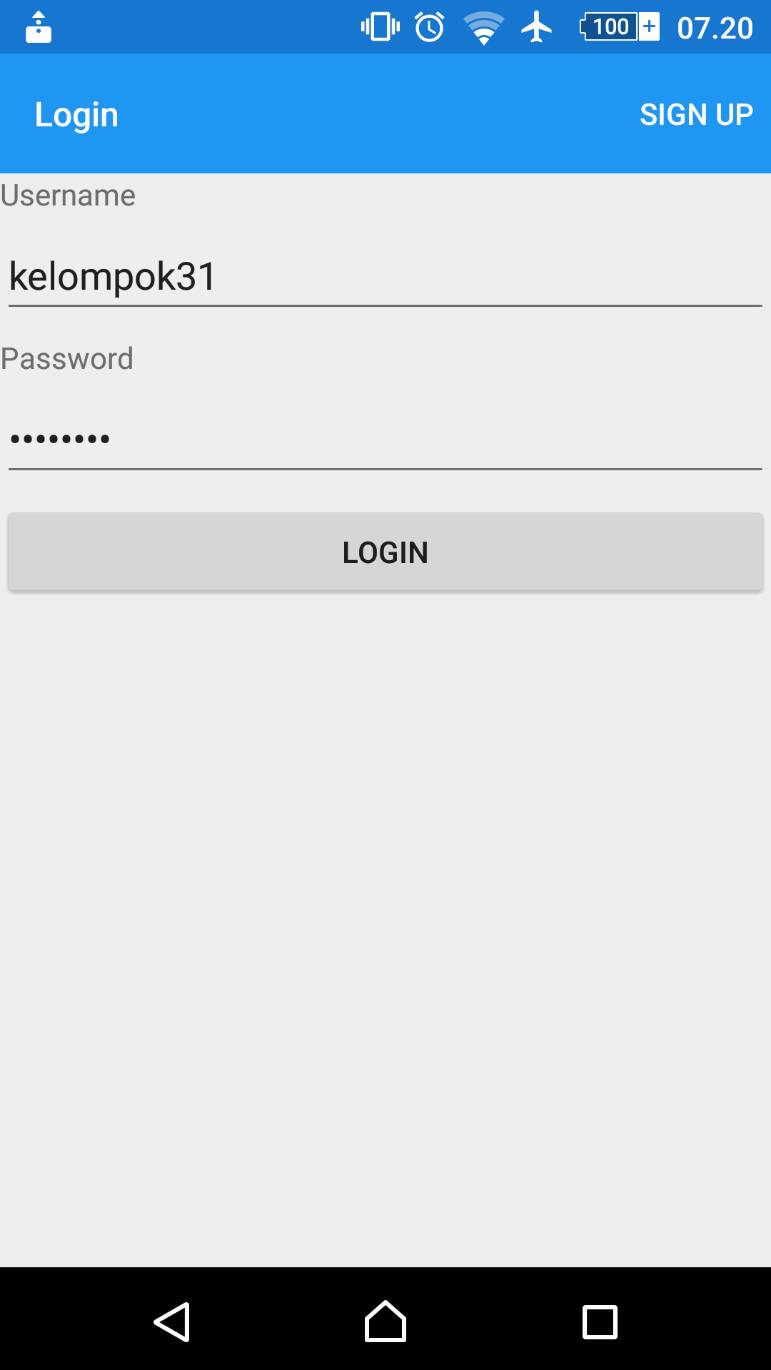
Gambar 2.26 Enable USB debugging Pada android

1. Ketika smartphone sudah terhubung melalui kabel USB dengan PC atau Laptop maka akan muncul dialog seperti berikut. Pilih OK, dan pastikan terhubung.
2. Kemudian sebelum membuild project maka lakukan Clean Solution terlebih dahulu pada Project Solution Kelompok31.
3. Apabila smartphone sudah terhubung dengan PC atau Laptop maka akan terdeteksi seperti pada gambar dibawah ini. Untuk melakukan build project sekaligus dploy project dapat dilakukan dengan langsung menekan tombol play.
4. Apabila aplikasi sudah sukses dibuild dan dideploy maka akan muncul tampilan seperti pada gambar dibawah ini.

a

Gambar 2.27 Hasil Program

* 1. **Analisa Hasil Percobaan**
     1. **Percobaan 1**

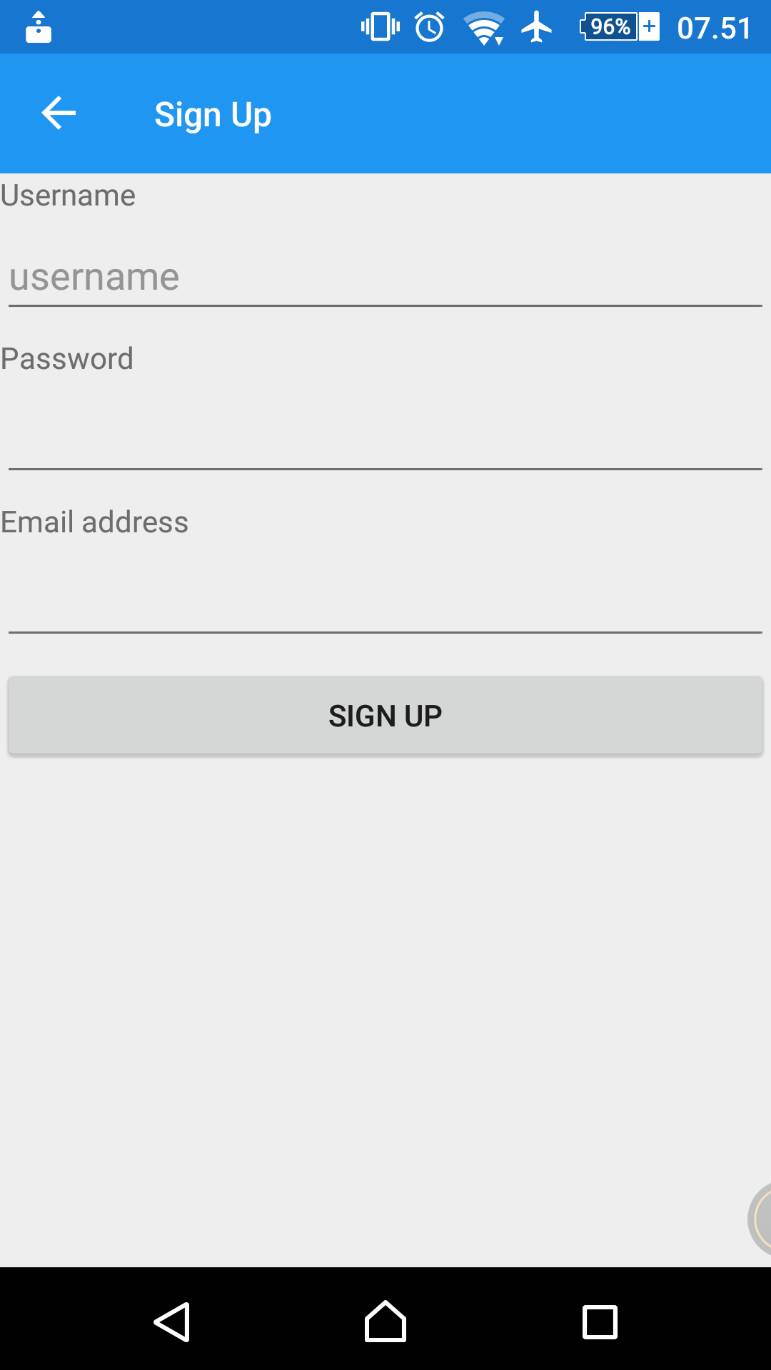


Gambar 2.28 Menu Tampilan Login

Pada hasil percobaan 1 adalah menu *login* yang mana dalam peletakan desain diatas diatur oleh file yang bernama LoginPage.xaml yang dimana komponen yang digunakan ialah 2 buah label, 2 buah *Text filed*, dan 1 *button* , dan *action* pada menu *login* tersebut. Pada LoginPage.xaml.cs yang dimana isi dari *file* ini terdiri dari instruksi yang digunakan untuk pengatur login karena tidak menggunakan database maka untuk username dan password diatur pada *folder* model yang bernama UserLogin.cs pada file ini berisi akun yang akan kita gunakan untuk *login* dengan *username* “Kelompok31” dan *Password* “password” . jika kita bisa berhasil masuk maka akan langsung masuk pada menu utama jika gagal makan aka nada pemberitahuan “Login Failed” .

|  |
| --- |
| if (isValid)  {  App.IsUserLoggedIn = true;  Navigation.InsertPageBefore(new HalamanUtama(), this);  await Navigation.PopAsync();  }  else  {  messageLabel.Text = "Login failed";  passwordEntry.Text = string.Empty;  }  }  bool AreCredentialsCorrect(User user)  {  return user.Username == UserLogin.Username && user.Password == UserLogin.Password;  } |

*Source code* diatas ialah pierintah / instruksi yang digunakan untuk proses masuk yang mana terletak pada LoginPage.xaml.cs.



Gambar 2.29 Tampilan SignUp

Pada tampilan menu diatur dalam *file* yang bernama SignUpPage.xaml yang dimana menggunakan *source code* yang membuat tampilan seperti diatas ialah :

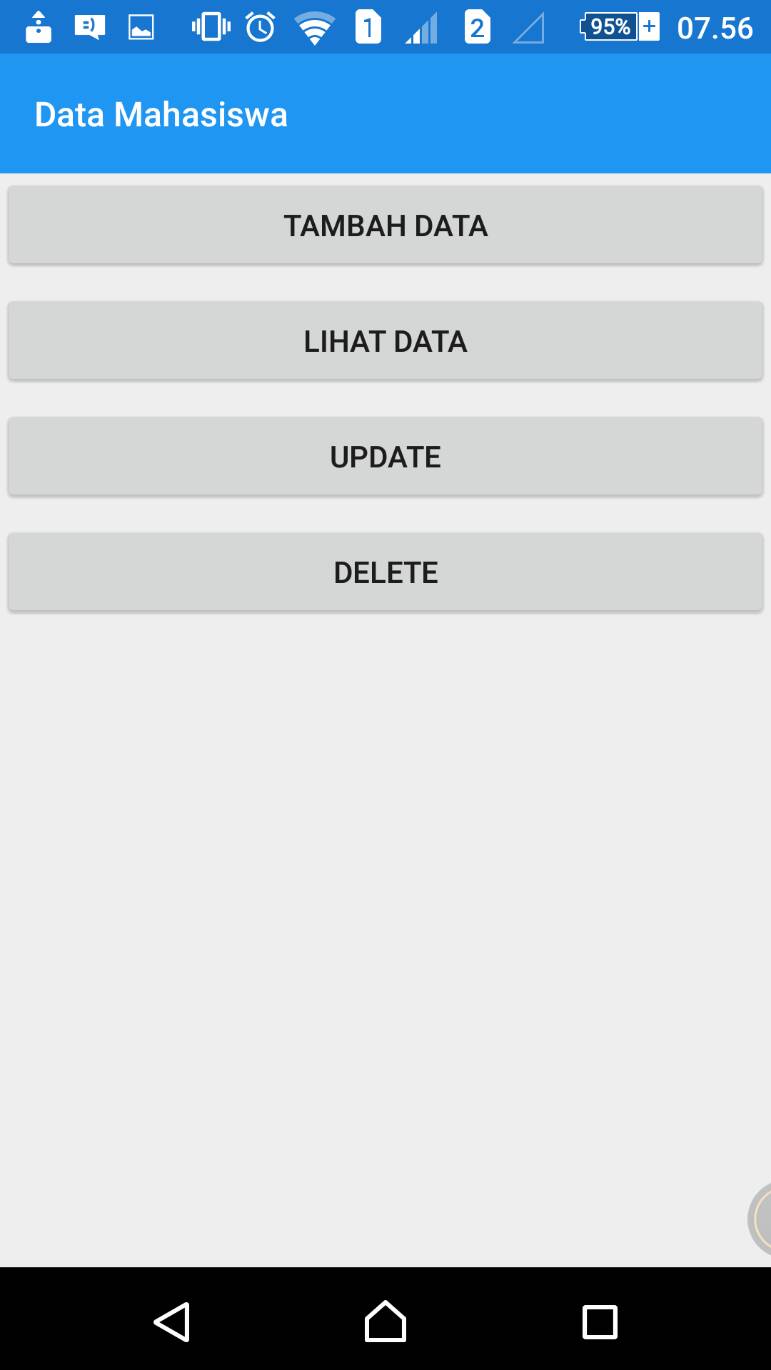
|  |
| --- |
| <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="Kelompok31.SignUpPage"  Title="Sign Up">  <ContentPage.Content>  <StackLayout VerticalOptions="StartAndExpand">  <Label Text="Username" />  <Entry x:Name="usernameEntry" Placeholder="username" />  <Label Text="Password" />  <Entry x:Name="passwordEntry" IsPassword="true" />  <Label Text="Email address" />  <Entry x:Name="emailEntry" />  <Button Text="Sign Up" Clicked="OnSignUpButtonClicked" />  <Label x:Name="messageLabel" />  </StackLayout>  </ContentPage.Content>  </ContentPage> |

Pada *source code* diatas menggunakan 2 *label* , 3 *TextField*, dan 1 *button* . pada menu SignUp ini untuk perintah button terdapat pada file SignUpPage.xaml.cs untuk *source code* utama dari tombol *button* ialah :

|  |
| --- |
| var user = new User()  {  Username = usernameEntry.Text,  Password = passwordEntry.Text,  Email = emailEntry.Text  };  // Sign up logic goes here  var signUpSucceeded = AreDetailsValid(user);  if (signUpSucceeded)  {  var rootPage = Navigation.NavigationStack.FirstOrDefault();  if (rootPage != null)  {  App.IsUserLoggedIn = true;  Navigation.InsertPageBefore(new MainPage(), Navigation.NavigationStack.First());  await Navigation.PopToRootAsync();  }  }  else  {  messageLabel.Text = "Sign up failed";  }  }  bool AreDetailsValid(User user)  {  return (!string.IsNullOrWhiteSpace(user.Username) && !string.IsNullOrWhiteSpace(user.Password) && !string.IsNullOrWhiteSpace(user.Email) && user.Email.Contains("@"));  } |

Disini terdapat coding *SignUp* berhasil bila mana ketiga *TextField* terisi dan sesuai dengan kententuan bila tidak dengan sesuai dengan ketentuan maka akan muncul perintah Label yang bertulis kan “*Sign Up Failed*” untuk tipe dari masukan yang kita isi pada ketiga *TextField* yang tersedia ialah *String*, bila berhasil maka akan dibawa pada Main Menu yang dimana hanya berisi tulisan "*Main app content goes here*" dan dan logout saja .

* + 1. **Percobaan 2**

Untuk Percobaan ke 2 kita menggunakan database *SQLite* yang dimana untuk penyimpanan utama nya terdapat pada File DataMahasiswa , untuk Halaman utama DataMahasiswa pada Percobaan ke2 seperti berikut :

Gambar 2.30 Menu Utama Data Mahasiswa

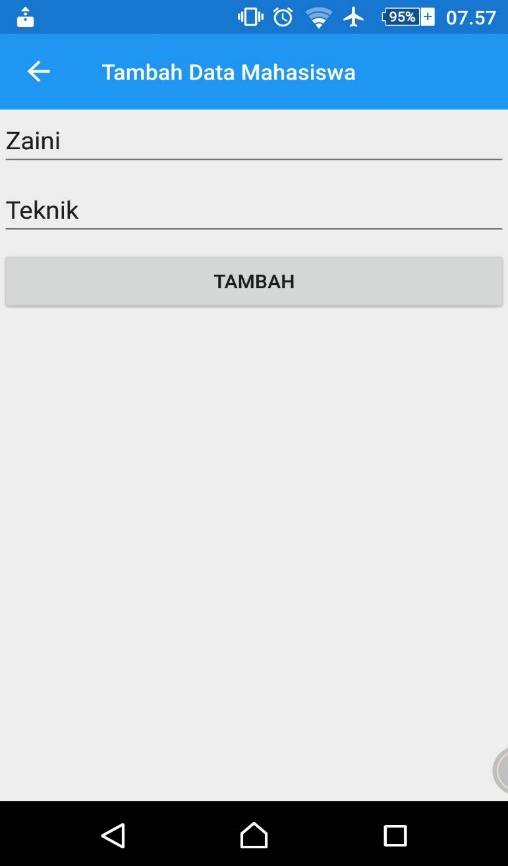
Menu ini terdapat 2 *button* yang mana fungsi *button* dan tampilan *button* nya diatur dalam file HalamanUtama.cs dimana untuk contoh *source code* nya :

|  |
| --- |
| Button button = new Button();  button.Text = "Tambah Data";  button.Clicked += Button\_Tambah\_Clicked;  stacklayout.Children.Add(button); |

Untuk penamaan pada tombol nya terdapat pada “button.text” untuk aksi nya akan memanggil “ Button\_Tambah\_Clicked” yang dimana isi dari fungsi nya berisi :

|  |
| --- |
| private async void Button\_Tambah\_Clicked(object sender, EventArgs e)  {  await Navigation.PushAsync(new HalamanTambahData());  } |

Instruksi ini berfungsi untuk membuka halam baru atau lebih tepat nya akan membuka HalamanTambahData, begitu juga untuk pembuatan dan fungsi pada tombol lihat data.



Gambar 2.31 Halaman Tambah Data

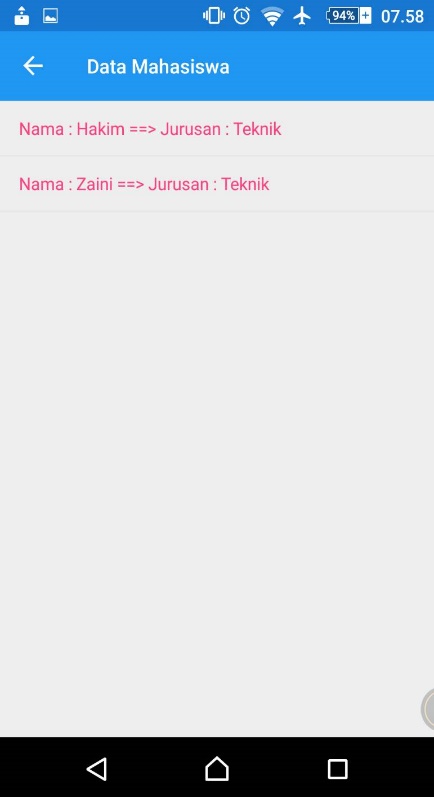
Pada menu ini terdapat pada folder view file HalamanTambahData.cs yang dimana pengaturan tombol dan fungsi nya terdapat pada file yang sama sama seperti diatas untuk tombol nya , untuk *TextField* nya :

|  |
| --- |
| \_nama = new Entry();  \_nama.Keyboard = Keyboard.Text;  \_nama.Placeholder = "Nama Mahasiswa";  stackLayout.Children.Add(\_nama); |

Untuk isi yang kita isikan akan tersimpan pada *variable* \_nama , pada menu ini kita menggunakan database SQLite untuk menyimpan data nya untuk fungsi utama pada tombol halaman ini berupa *source code* :

|  |
| --- |
| var db = new SQLiteConnection(\_dbPath);  db.CreateTable<DataMahasiswa>();  var maxPk = db.Table<DataMahasiswa>().OrderByDescending(c => c.Id).FirstOrDefault();  DataMahasiswa dbm = new DataMahasiswa()  {  Id = (maxPk == null ? 1 : maxPk.Id + 1),  Nama = \_nama.Text,  Jurusan = \_jurusan.Text  };  db.Insert(dbm);  await DisplayAlert(null, "Data " + dbm.Nama + " Berhasil Disimpan", "Ok");  await Navigation.PopAsync(); |

Pada *source code* diatas membuat koneksi SQLite dan akan membuat table pada DataMahasiswa untuk ID nya akan otomatis akan bertambah untuk isi dari nama dan jurusan nya mengambul dari *TextField* \_nama.Text dan juga \_jurusan.Text dan kemudian akan disimpan pada databse .



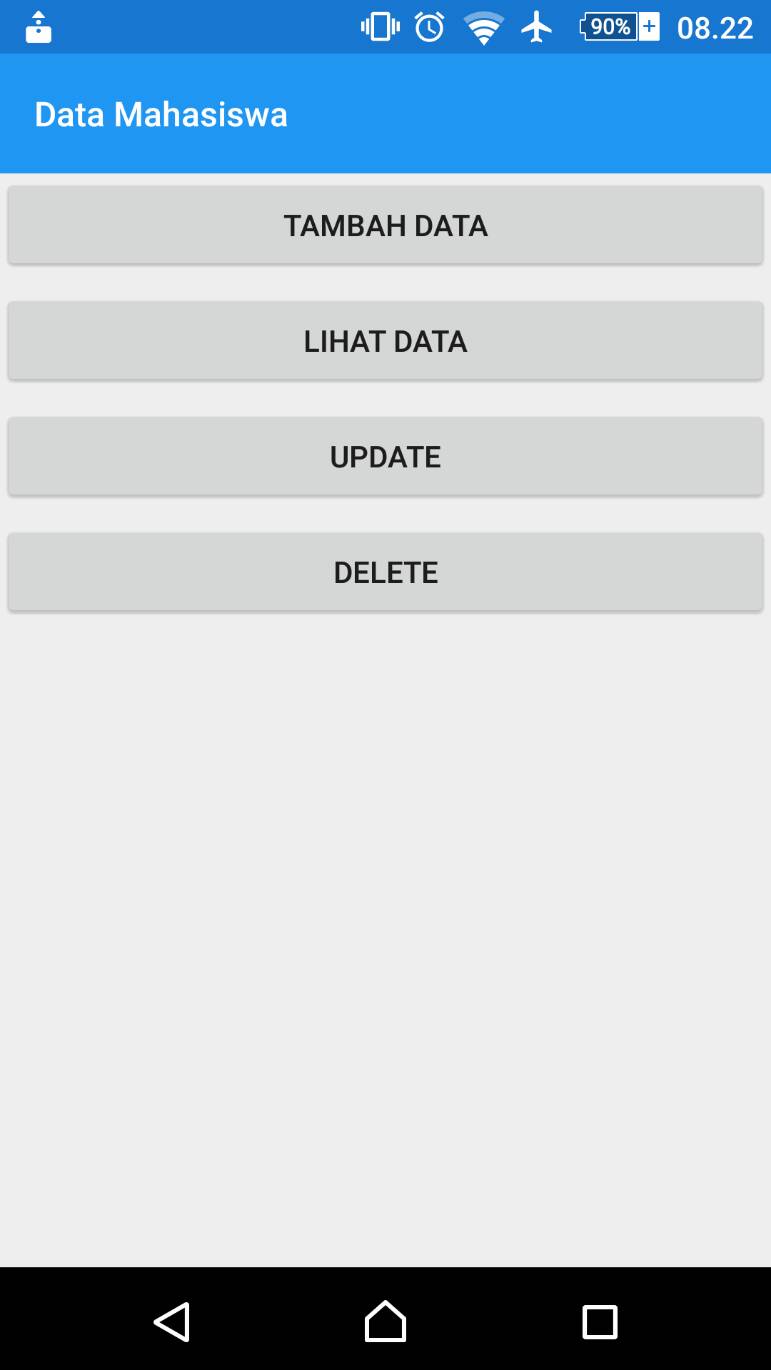
Gambar 2.32 Halaman Lihat Data

Untuk tampilan halaman lihat data terdapat pada folder *view* file HalamanLihatData.cs yang mana halaman ini berisi hasil dari tambah data yang telah kita lakukan pada saati kita memberi tambah data . Untuk *source code* fungsi utama nya :

|  |
| --- |
| this.Title = "Data Mahasiswa";  var db = new SQLiteConnection(\_dbPath);  StackLayout stackLayout = new StackLayout();  \_listView = new ListView();  \_listView.ItemsSource = db.Table<DataMahasiswa>().OrderBy(x => x.Nama).ToList();  stackLayout.Children.Add(\_listView);  Content = stackLayout; |

Pada *source code* nya akan menampilkan list data yang kita imputkan dan akan ditampilkan per *item* nya dengan menggunakan Database SQLite tentu nya .

* 1. **Tugas**



Gambar 2.33 Halaman Menu Utama Tugas

Pada halaman utama tugas ini terdapat penambahan 2 buah tombol yang mana tombol Edit dan hapus untuk *source code* nya terpadat pada folder *view* file HalamanUtama.cs dan untuk menambahakan tombol kedua nya seprti berikut :

|  |
| --- |
| button = new Button();  button.Text = "Edit";  button.Clicked += Button\_Edit\_Clicked;  stacklayout.Children.Add(button);  button = new Button();  button.Text = "Hapus";  button.Clicked += Button\_Hapus\_Clicked;  stacklayout.Children.Add(button); |

Pembuatan button dan aksi yang akan dilakukan bila button itu dipencet ialah fungsi Button\_Edit\_Clicked dan Button\_Hapus\_Clicked yang dimana akan membuka halaman Hapus data dan Edit Data *source code* nya :

|  |
| --- |
| private async void Button\_Edit\_Clicked(object sender, EventArgs e)  {  await Navigation.PushAsync(new HalamanEditData());  }  private async void Button\_Hapus\_Clicked(object sender, EventArgs e)  {  await Navigation.PushAsync(new HalamanHapusData());  } |

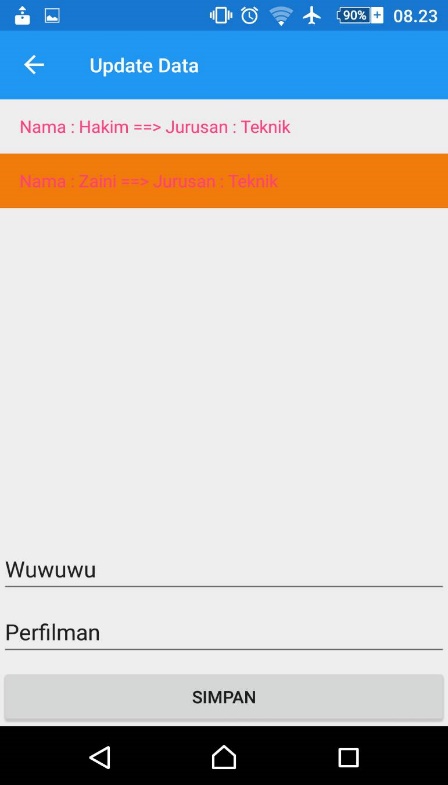
Lalu kita membuat new item pada folder view kita dengan nama HalamanEditData.cs kemudian masukan source code berikut :

|  |
| --- |
| using Kelompok31.Model;  using SQLite;  using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Text;  using Xamarin.Forms;  namespace Kelompok31.View  {  public class HalamanEditData : ContentPage  {  private ListView \_listView;  private Entry \_idEntry;  private Entry \_namaEntry;  private Entry \_jurusanEntry;  private Button \_button;  DataMahasiswa \_data = new DataMahasiswa();  string \_dbPath = Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), "myDB.db4");  public HalamanEditData ()  {  this.Title = "Edit Data Mahasiswa";  var db = new SQLiteConnection(\_dbPath);  StackLayout stackLayout = new StackLayout();  \_listView = new ListView();  \_listView.ItemsSource = db.Table<DataMahasiswa>().OrderBy(x => x.Nama).ToList();  \_listView.ItemSelected += \_listView\_ItemSelected;  stackLayout.Children.Add(\_listView);  \_idEntry = new Entry();  \_idEntry.Placeholder = "ID";  \_idEntry.IsVisible = false;  stackLayout.Children.Add(\_idEntry);  \_namaEntry = new Entry();  \_namaEntry.Keyboard = Keyboard.Text;  \_namaEntry.Placeholder = "Nama Anda";  stackLayout.Children.Add(\_namaEntry);  \_jurusanEntry = new Entry();  \_jurusanEntry.Keyboard = Keyboard.Text;  \_jurusanEntry.Placeholder = "Jurusan";  stackLayout.Children.Add(\_jurusanEntry);  \_button = new Button();  \_button.Text = "Edit";  \_button.Clicked += \_button\_Clicked;  stackLayout.Children.Add(\_button);  Content = stackLayout;  }  private async void \_button\_Clicked(object sender, EventArgs e)  {  var db = new SQLiteConnection(\_dbPath);  DataMahasiswa data = new DataMahasiswa()  {  Id = Convert.ToInt32(\_idEntry.Text),  Nama = \_namaEntry.Text,  Jurusan = \_jurusanEntry.Text  };  db.Update(data);  await Navigation.PopAsync();  }  private void \_listView\_ItemSelected(object sender, SelectedItemChangedEventArgs e)  {  \_data = (DataMahasiswa)e.SelectedItem;  \_idEntry.Text = \_data.Id.ToString();  \_namaEntry.Text = \_data.Nama;  \_jurusanEntry.Text = \_data.Jurusan;  }  }  } |

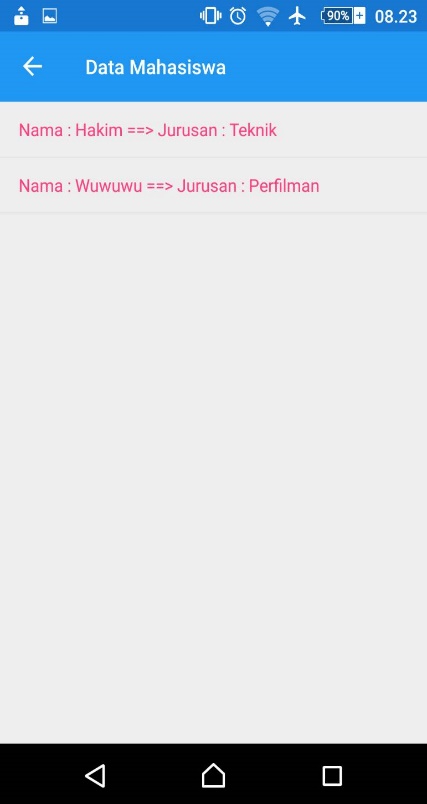
Untuk bagian fungsi terpenting pada halaman edit ini ialah edit data nya yang mana menggunakan *source code* :

|  |
| --- |
| private async void \_button\_Clicked(object sender, EventArgs e)  {  var db = new SQLiteConnection(\_dbPath);  DataMahasiswa data = new DataMahasiswa()  {  Id = Convert.ToInt32(\_idEntry.Text),  Nama = \_namaEntry.Text,  Jurusan = \_jurusanEntry.Text  };  db.Update(data);  await Navigation.PopAsync();  }  private void \_listView\_ItemSelected(object sender, SelectedItemChangedEventArgs e)  {  \_data = (DataMahasiswa)e.SelectedItem;  \_idEntry.Text = \_data.Id.ToString();  \_namaEntry.Text = \_data.Nama;  \_jurusanEntry.Text = \_data.Jurusan;  } |

Untuk pada \_listView\_ItemSelected bila list dari data tersebut dipilih maka akan terisi pada *texfiled* \_namaEntry dan \_jurusanEntry untuk id pada data tersbut tidak ditampilkan. Pada *source code* \_button\_clicked akan mengupdate data pada *TextField* \_namaEntry dan \_jurusanEntry dengan perintah “db.Update(data);” maka data akan diupdate dan tersimpan pada data base , untuk tampilan proses pengeditan dan hasil nya seprti berikut :



Gambar 2.34 Tampilan Edit Data sebelum Diedit



Gambar 2.35 Hasil Setelah Diedit

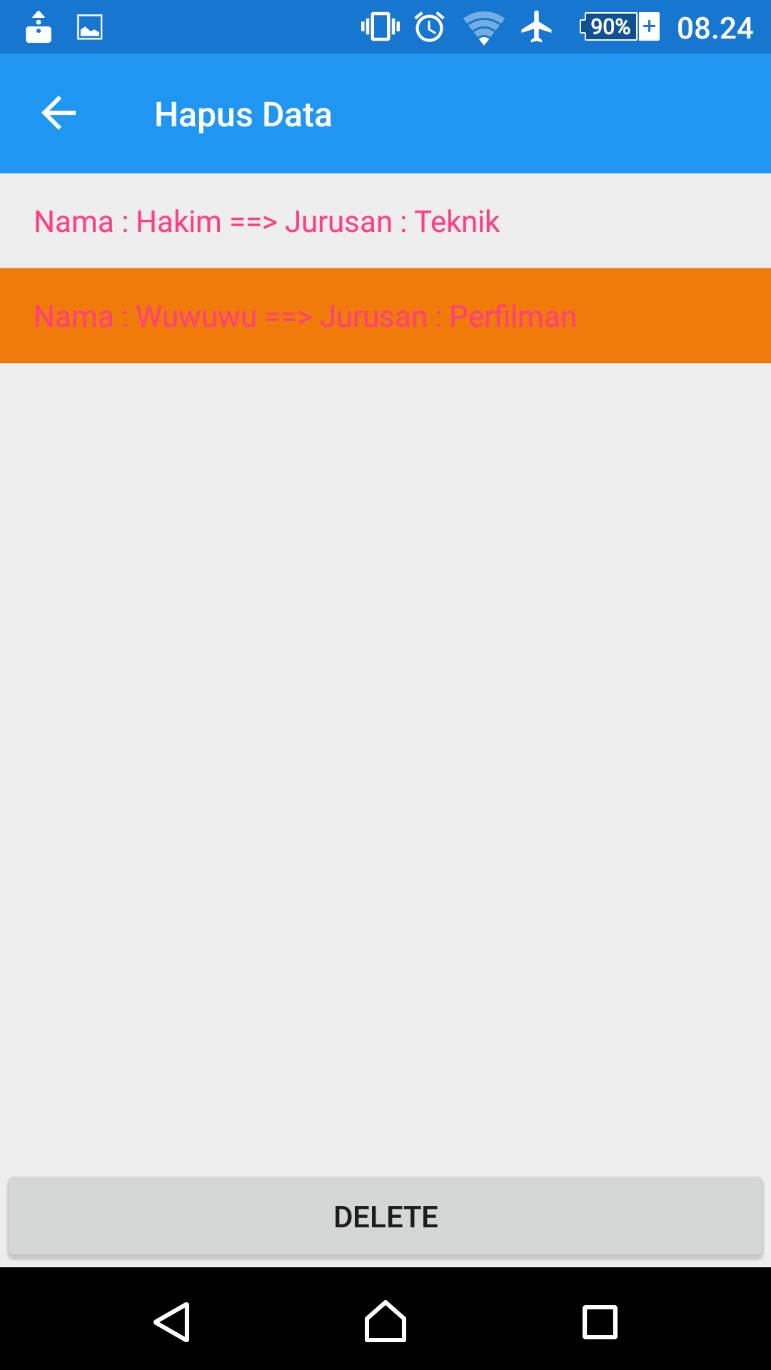
Kemudian membuat new item pada folder view yang bernama HalamanHapusData.cs . untuk *source code* nya :

|  |
| --- |
| using Kelompok31.Model;  using SQLite;  using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Text;  using Xamarin.Forms;  namespace Kelompok31.View  {  public class HalamanHapusData : ContentPage  {  private ListView \_listView;  private Button \_button;  DataMahasiswa \_data = new DataMahasiswa();  string \_dbPath = Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), "myDB.db4");  public HalamanHapusData ()  {  this.Title = "Hapus Data Mahasiswa";  var db = new SQLiteConnection(\_dbPath);  StackLayout stackLayout = new StackLayout();  \_listView = new ListView();  \_listView.ItemsSource = db.Table<DataMahasiswa>().OrderBy(x => x.Nama).ToList();  \_listView.ItemSelected += \_listView\_ItemSelected;  stackLayout.Children.Add(\_listView);  Content = stackLayout;  \_button = new Button();  \_button.Text = "Hapus";  \_button.Clicked += \_button\_Clicked;  stackLayout.Children.Add(\_button);  }  private void \_listView\_ItemSelected(object sender, SelectedItemChangedEventArgs e)  {  \_data = (DataMahasiswa)e.SelectedItem;  }  private async void \_button\_Clicked(object sender, EventArgs e)  {  var db = new SQLiteConnection(\_dbPath);  db.Table<DataMahasiswa>().Delete(x => x.Id == \_data.Id);  await Navigation.PopAsync();  }  }  } |

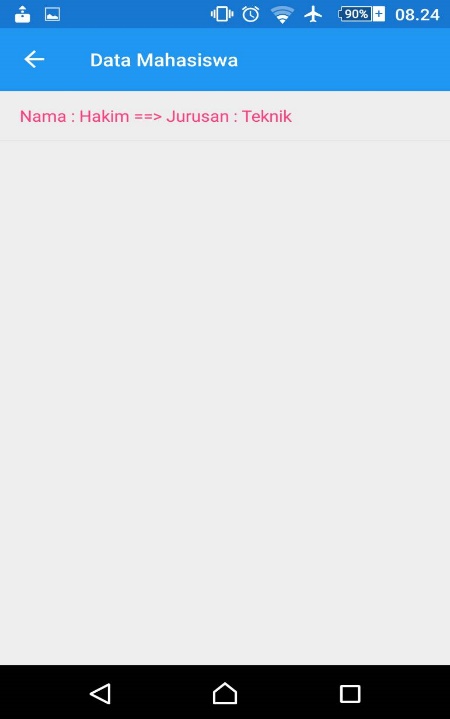
Untuk *design* halaman , fungsi-fungsi dari tombol , dan list dari database nya semua tertulis di *source code* diatas bagian yang lain nya sama saja dengan halaman edit data yang membedakan disini ialah fungsi dari *button* nya yang dimana tombol dari button nya akan menghapus file yang kita pilih dari data base. Fungsi dari *button* itu sendiri :

|  |
| --- |
| var db = new SQLiteConnection(\_dbPath);  db.Table<DataMahasiswa>().Delete(x => x.Id == \_data.Id);  await Navigation.PopAsync(); |

Data akan dihapus melalui database dengan parameter ID dari data tersebut . untuk tampilan dan proses setelah dan sesudah hapus data sebagai berikut :



Gambar 2.36 Halaman Hapus Data, Sebelum data dihapus



Gambar 2.37 Lihat Data setelah data dihapus

(Link Github Tugas Kelompok31 : https://github.com/malikhakim/Xamarin)

* 1. **Kesimpulan**
  2. Db.Table.NamaFileDatabase.Delete(Kondisi) Berguna untuk menghapus data *table* pada *database*.
  3. Db.Update(data) berguna untuk meng*update* data pada *database*.
  4. private async void \_button\_Clicked(object sender, EventArgs e) berfungsi sebagai aksi pada saat Tombol ditekan.
  5. \_listView.ItemsSource=db.Table<Lokasi>().OrderBy(kondisi).ToList(); untuk menampilka *list* data secara berurutan pada *database*.
  6. SQLiteconnection berfungsi sebagai penghubung *file* dengan *database* yang dibuat.
  7. Versi Dari Xamarin.Forms harus sesuai dengan *Platform* kita supaya tidak terjadi data yang hilang.
  8. Penambahan *Toolbar* item pada Xamarin bisa diatur untuk ditekan dan melakukan aksi.