

## **BAB II**

### **Xamarin**

#### **2.1 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 mampu menerapkan penggunaan Xamarin dan *database* SQLite dalam pembuatan program sederhana.

## 2.2 Alat dan Bahan

### 2.2.1 PC/Notebook

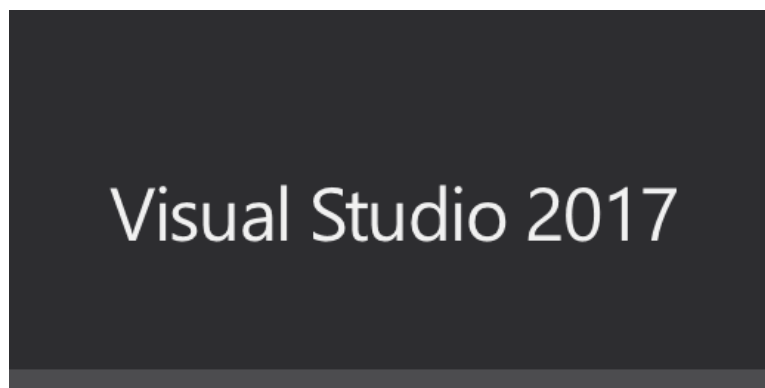
PC/Notebook digunakan untuk menginstall aplikasi Visual Studio Community 2017 dengan spesifikasi minimal yaitu memiliki RAM 4 GB. Hal ini bertujuan agar praktikum dapat berjalan dengan lancar.



Gambar 2.1 PC/Notebook

### 2.2.2 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

### 2.2.3 Device Android dan Kabel Data

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



Gambar 2.3 Device Android dan Kabel Data USB

#### **2.2.4 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 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, SDK, Integrated Development Environment (IDE), dan dokumentasi (umumnya berupa MSDN Library). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe.

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 di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework).

(Sumber : Modul 1 BAB II Xamarin – 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 menggunakan 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 BAB II Xamarin – 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:

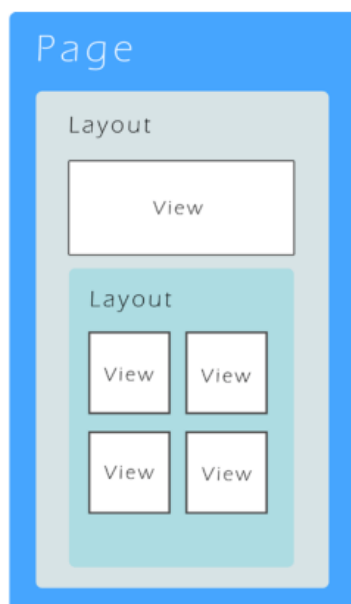
- **Portable:** berisi project utama yang akan dishare secara otomatis ke project yang mempunyai platform spesifik.
- **Droid:** project untuk platform Android.
- **IOS:** project untuk platform IOS.



## 2. 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

## 3. 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

#### 4. 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 – scrollable 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

#### 5. 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.

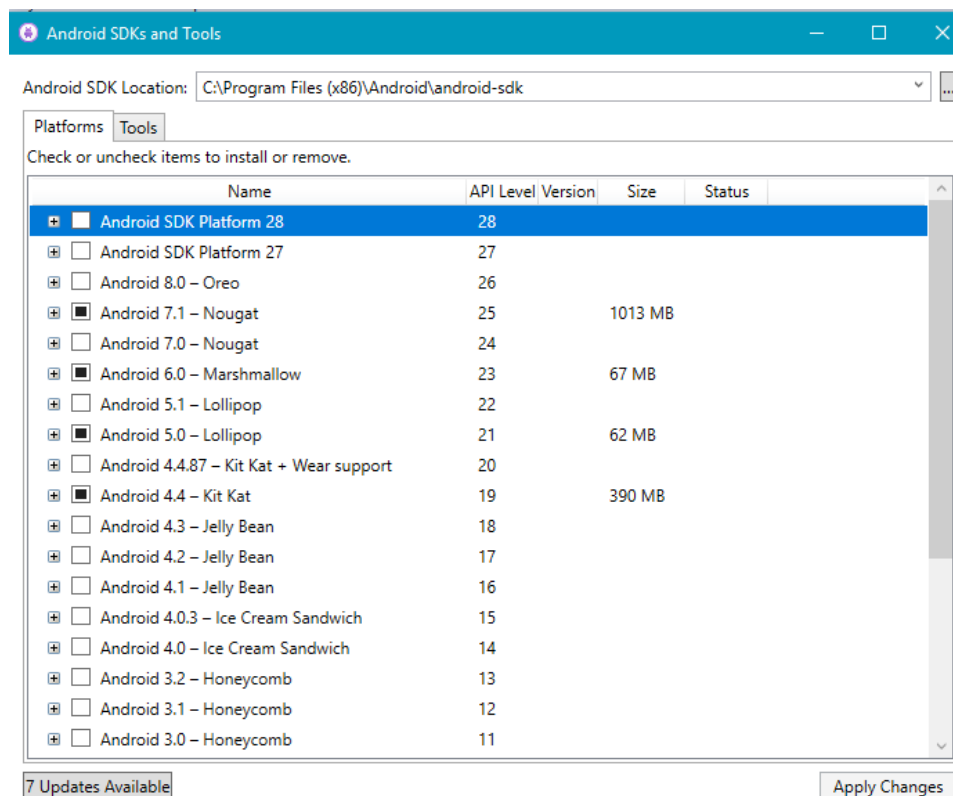


- **RelativeLayout:** pengaturan posisi kontrol berdasarkan kontrol yang lain.
- **Grid:** membuat layout yang terdiri dari kolom dan baris seperti tabel.

(Sumber : Modul 1 BAB II Xamarin – Praktikum MDP 2018)

### 2.3.4 SDK Pada Xamarin

Android menggunakan beberapa pengaturan level API Android untuk menentukan kompatibilitas aplikasi yang dibangun 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 BAB II Xamarin – 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 BAB II Xamarin – Praktikum MDP 2018)

### 2.3.6 SQLite

SQLite merupakan sebuah sistem manajemen basisdata relasional yang bersifat ACID-compliant dan memiliki ukuran pustaka kode yang relatif kecil, ditulis dalam bahasa C. SQLite merupakan proyek yang bersifat public domain yang dikerjakan oleh D. Richard Hipp.

Tidak seperti pada paradigma client-server 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 secara langsung melalui bahasa pemrograman. Mekanisme seperti ini tentunya membawa keuntungan karena dapat mereduksi *overhead*, *latency times*, dan secara keseluruhan lebih sederhana.

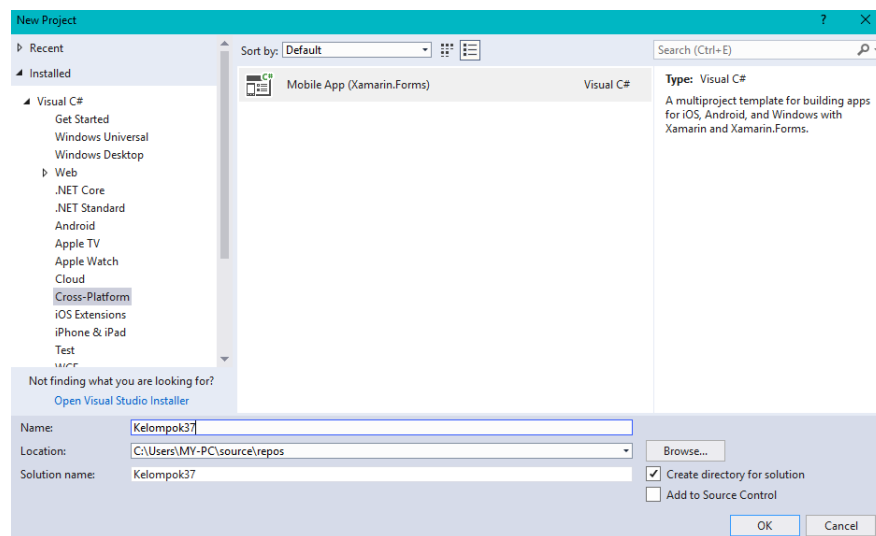
Seluruh elemen basisdata (definisi data, tabel, indeks, dan data) disimpan sebagai sebuah *file*. Kesederhanaan dari sisi disain tersebut bisa diraih dengan cara mengunci keseluruhan *file* basis data pada saat sebuah transaksi dimulai.

(Sumber : Modul 1 BAB II Xamarin – Praktikum MDP 2018)

## 2.4 Langkah Kerja

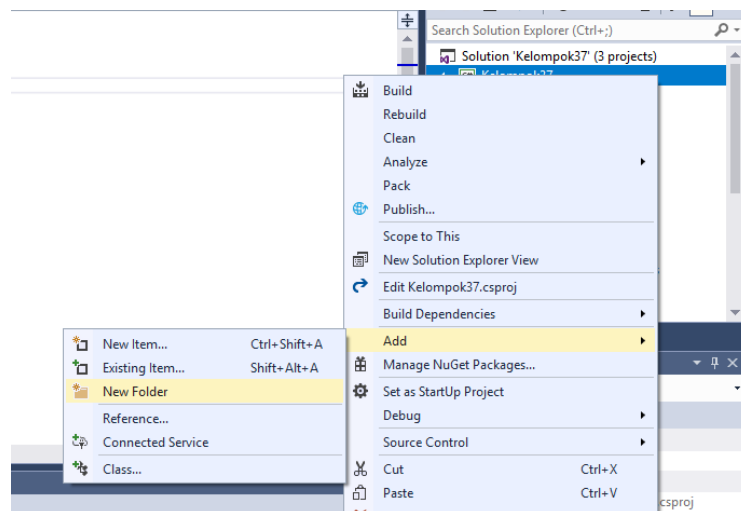
### 2.4.1 Membuat Login

1. Membuat project baru dengan nama Kelompok37. Pilih Cross-Platform App (Xamarin.Forms).



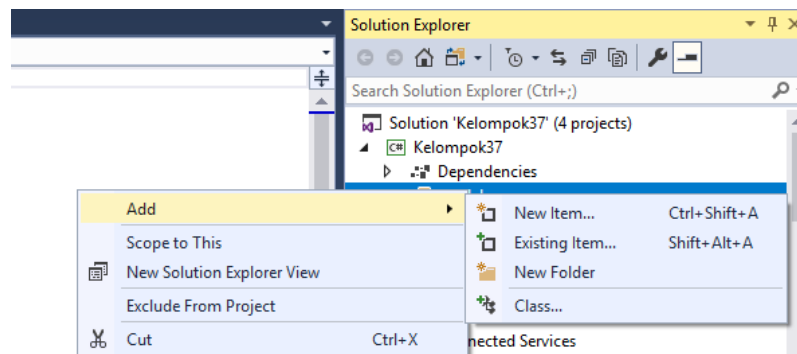
Gambar 2.10 Membuat Project baru

2. Membuat folder View dan Model pada project Xamarin.Forms Kelompok37. Klik kanan pada project Kelompok37 > Add > New Folder

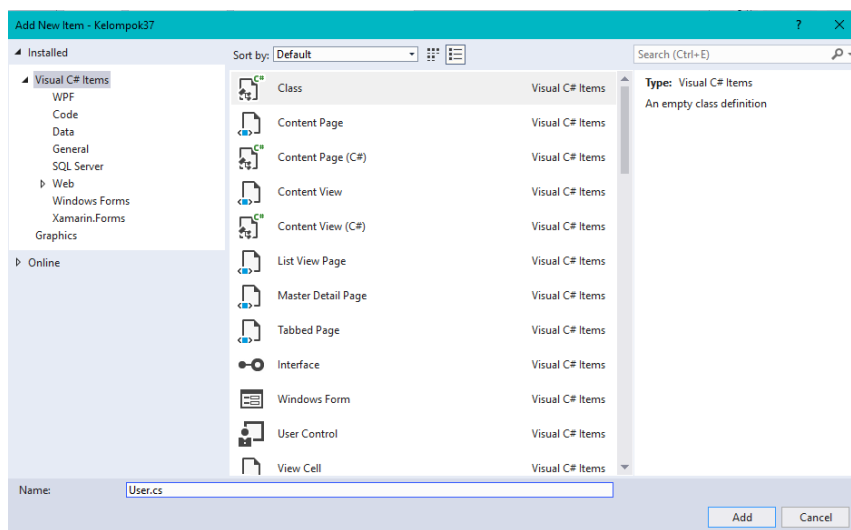


Gambar 2.11 Membuat folder view dan model

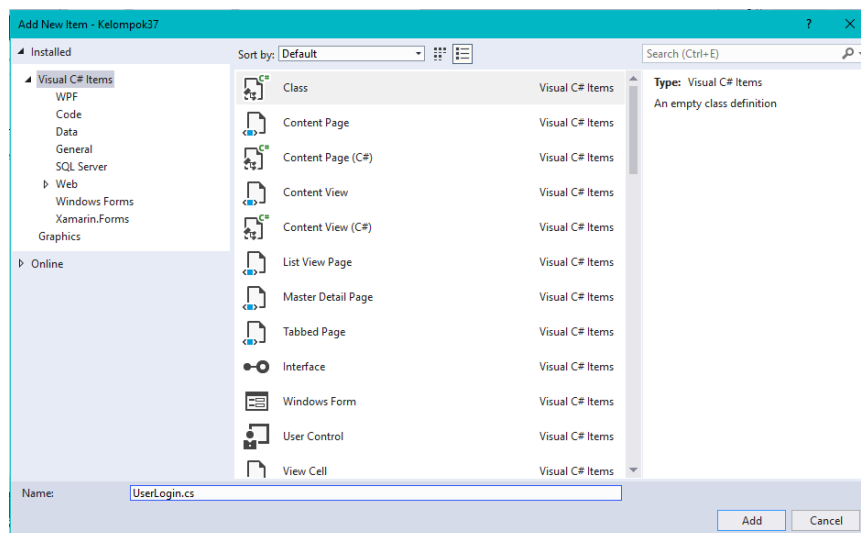
3. Membuat Class User.cs dan UserLogin.cs pada direktori atau folder model yang sudah dibuat sebelumnya. Klik kanan pada folder Model > Add > Class



Gambar 2.12 Cara menambahkan class



Gambar 2.13 Membuat class User.cs



Gambar 2.14 Membuat class UserLogin.cs

4. 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 Kelompok37.Model
{
    public class User
    {
        public string Username { get; set; }

        public string Password { get; set; }

        public string Email { get; set; }
    }
}

```

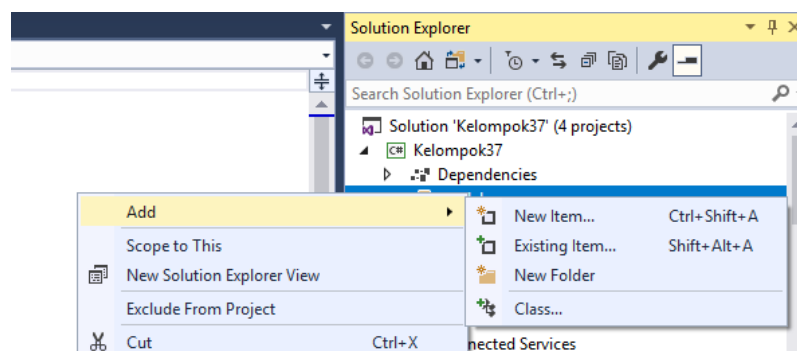
5. Masukkan source code berikut ini ke dalam *file* UserLogin.cs. Pada *file* ini terdapat dua variabel yang digunakan sebagai user dan password ketika akan melakukan login ke dalam aplikasi.

```

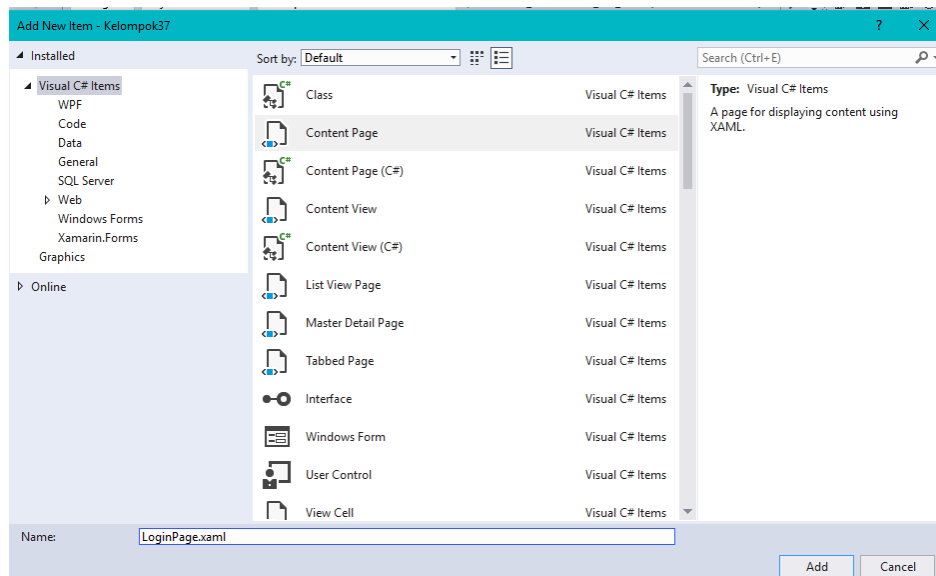
namespace Kelompok37
{
    public static class UserLogin
    {
        public static string Username = "Kelompok37";
        public static string Password = "password";
    }
}

```

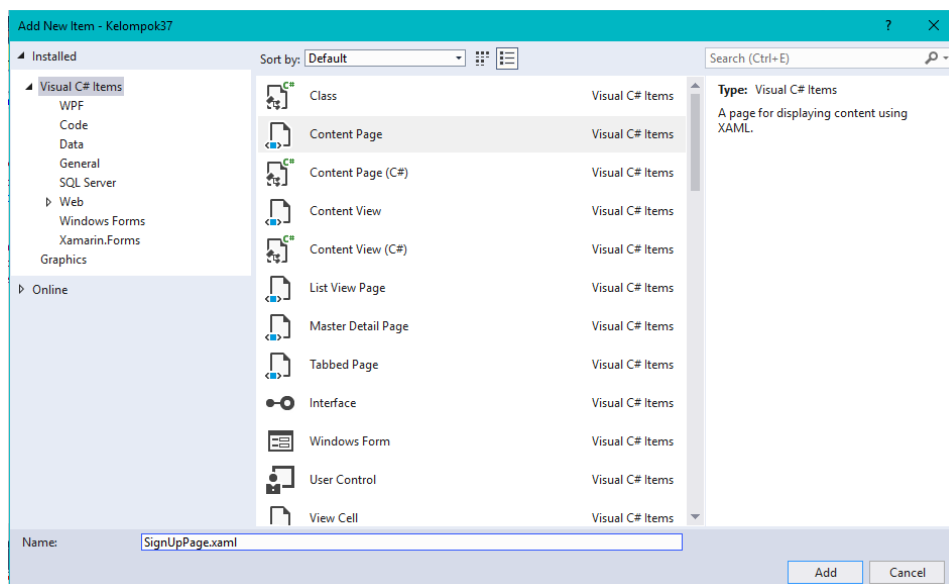
6. Membuat Content Page, masing – masing *file* diberi nama LoginPage.xaml, SignUpPage.xaml. Pada *file* Content Page terdapat dua jenis *file* sekaligus yaitu *file* dengan format .xaml dan .cs. Klik kanan pada project Kelompok37 > Add > New item > Content Page



Gambar 2.15 Cara menambahkan New Item



Gambar 2.16 Menambahkan Content Page LoginPage.xaml



Gambar 2.17 Menambahkan Content Page SignUpPage.xaml

## 7. 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="Kelompok37.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" />
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```

```

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>

```

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

```

using Kelompok37.Model;
using Kelompok37.View;
using System;
using Xamarin.Forms;

namespace Kelompok37
{
    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;
        }
    }
}

```

9. 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="Kelompok37.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>

```

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

```

using Kelompok37.Model;
using System;
using System.Linq;
using Xamarin.Forms;

namespace Kelompok37
{
    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.IsNullOrEmpty(user.Username) &&
!string.IsNullOrEmpty(user.Password) &&
!string.IsNullOrEmpty(user.Email) &&
user.Email.Contains("@"));
    }
}

```

11. 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="Kelompok37.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>

```

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

```

using Kelompok37;
using System;
using Xamarin.Forms;

```

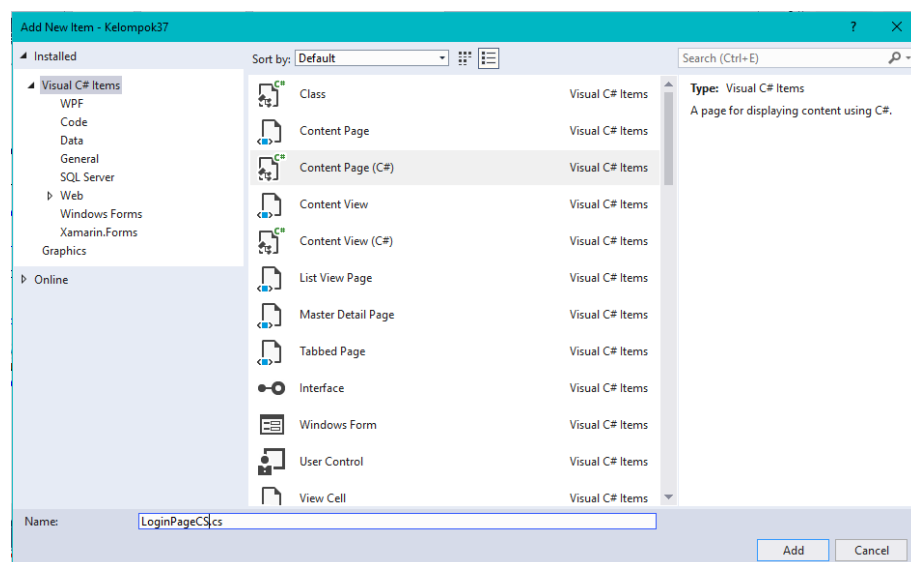
```

namespace Kelompok37
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            InitializeComponent();

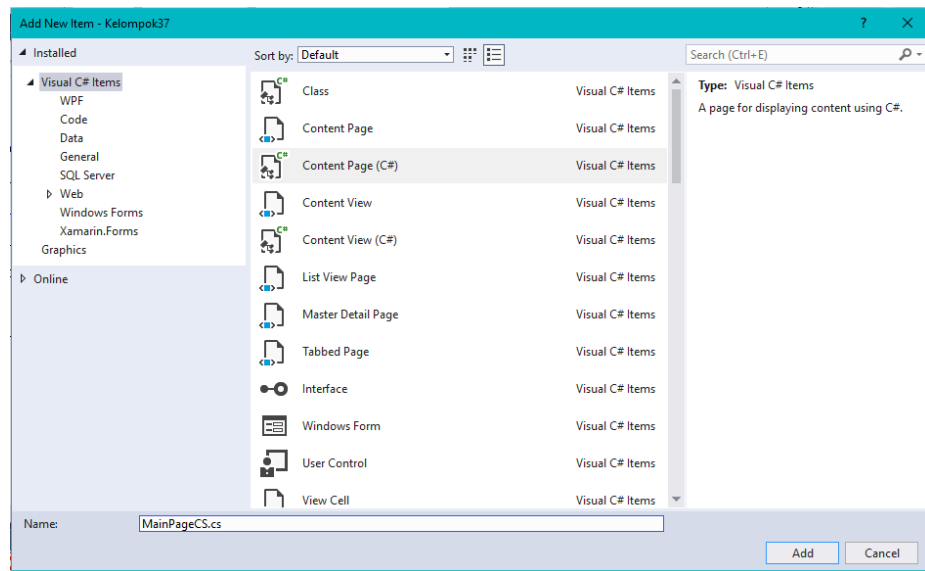
            async void OnLogoutButtonClicked(object sender,
            EventArgs e)
            {
                App.IsUserLoggedIn = false;
                Navigation.InsertPageBefore(new LoginPage(),
                this);
                await Navigation.PopAsync();
            }
        }
    }
}

```

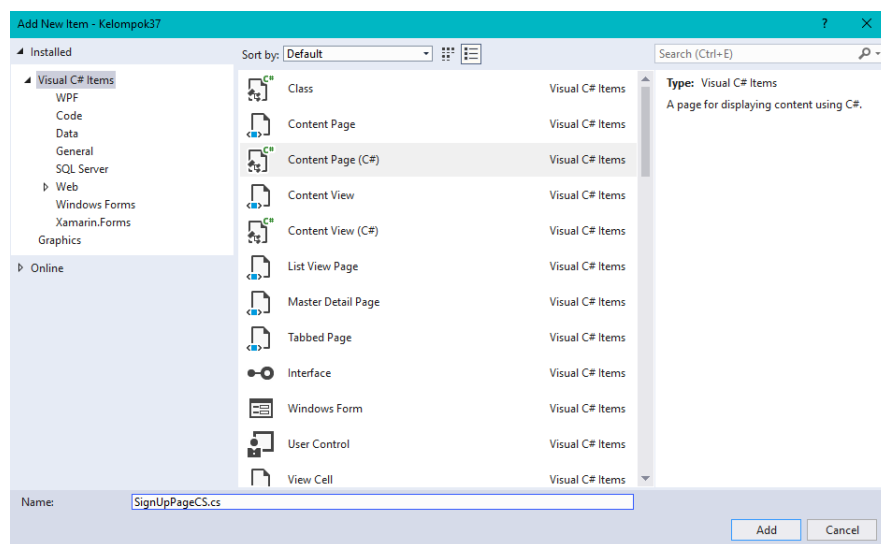
13. Kemudian langkah selanjutnya yaitu membuat Content Page (C#), masing-masing *file* dibuat dan diberi nama dengan LoginPageCS.cs, MainPageCS.cs dan SignUpPageCS.cs. Klik kanan pada project Kelompok37 > Add > New item > Content Page (C#)



Gambar 2.18 Membuat Content Page (C#) LoginPageCS.cs



Gambar 2.19 Membuat Content Page (C#) MainPageCS.cs



Gambar 2.20 Membuat Content Page (C#) SignUpPageCS.cs

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

```
using Kelompok37.Model;
using System;
using Xamarin.Forms;

namespace Kelompok37
{
    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;
}
}

```

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

```

using System;
using Xamarin.Forms;

namespace Kelompok37
{
    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();
    }
}

```

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

```

using Kelompok37;
using Kelompok37.Model;
using System;
using System.Linq;
using Xamarin.Forms;

namespace Kelompok37
{
    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

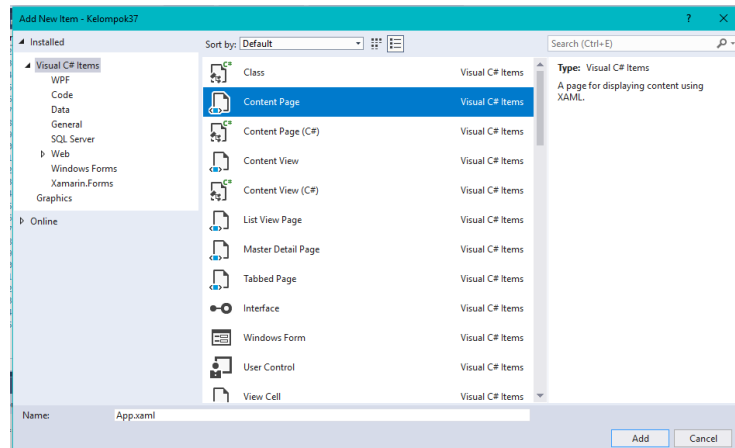
            var signUpSucceeded = AreDetailsValid(user);
            if (signUpSucceeded)
            {
                var rootPage =
Navigation.NavigationStack.FirstOrDefault();
                if (rootPage != null)
                {
                    App.IsUserLoggedIn = true;
                    Navigation.InsertPageBefore(new
MainPageCS(), Navigation.NavigationStack.First());
                    await Navigation.PopToRootAsync();
                }
            }
            else
            {
                messageLabel.Text = "Sign up failed";
            }
        }

        bool AreDetailsValid(User user)
        {
            return
(!string.IsNullOrEmpty(user.Username) &&
!string.IsNullOrEmpty(user.Password) &&
!string.IsNullOrEmpty(user.Email) &&
user.Email.Contains("@"));
        }
    }
}

```

17. Membuat content page dengan nama App.xaml dengan cara klik kanan project Kelompok37 > Add > New Item > Content Page





Gambar 2.21 Membuat Content Page App.xaml

## 18. Isikan source code berikut pada App.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Kelompok37.View;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

[assembly: XamlCompilation(XamlCompilationOptions.Compile)]
namespace Kelompok37
{
    public partial class App : Application
    {
        public static bool IsUserLoggedIn { get; internal
set; }
        public App()
        {
            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()
```

```

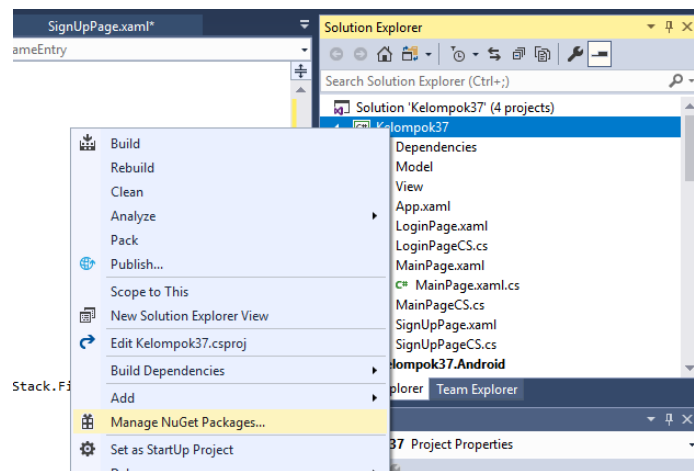
    {
        // Handle when your app sleeps
    }

    protected override void OnResume()
    {
        // Handle when your app resumes
    }
}

```

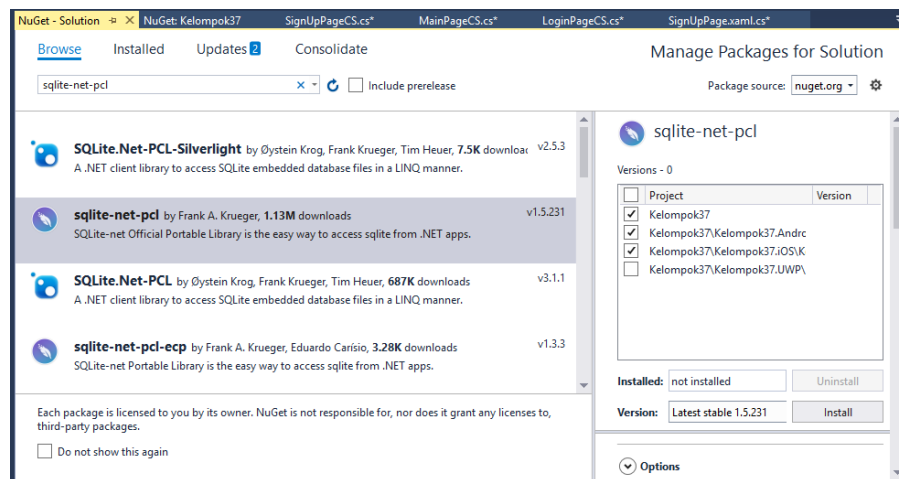
### 2.4.2 Percobaan SQLite

1. Klik kanan pada project Solution Kelompok37 kemudian pilih menu Manage NuGet Packages for Solution.



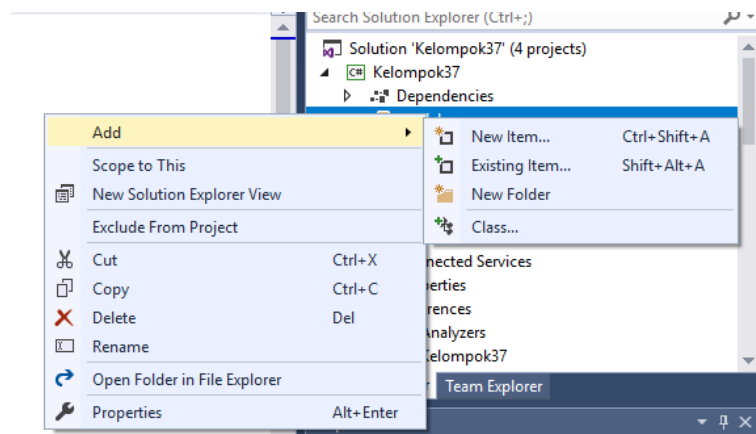
Gambar 2.22 Manage NuGet Packages

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*.

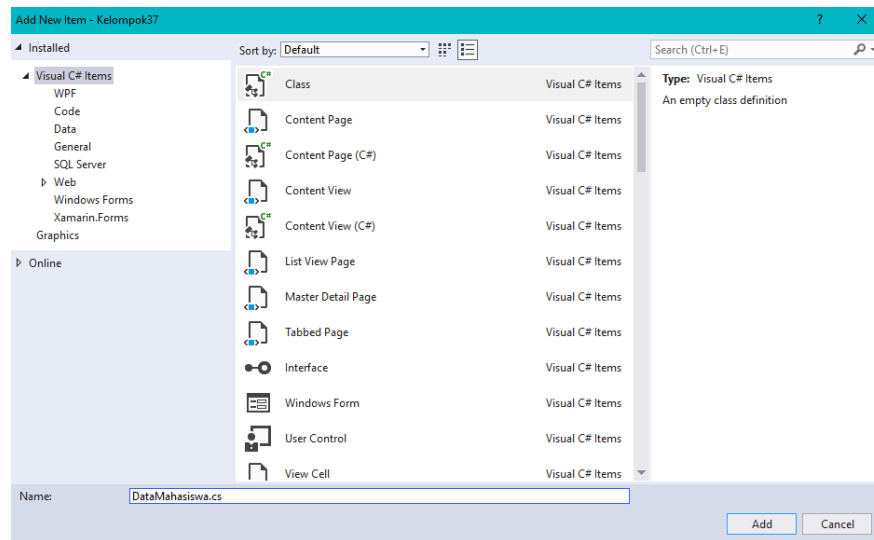


Gambar 2.23 Instalasi NuGet Packages

3. 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 2.24 Menambahkan Class baru pada folder Model



Gambar 2.25 Membuat class DataMahasiswa.cs

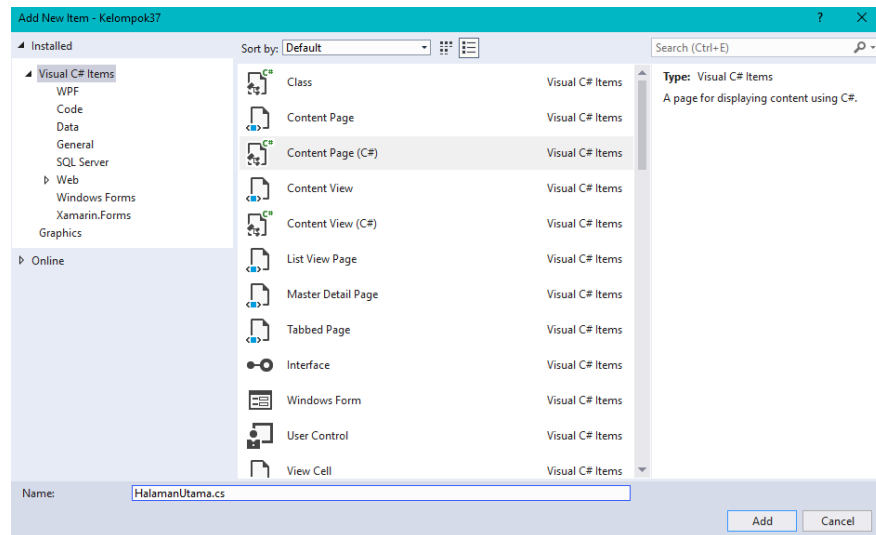
4. Ketik *source code* berikut ke dalam DataMahasiswa.cs

```
using SQLite;
using System;
using System.Collections.Generic;
using System.Text;

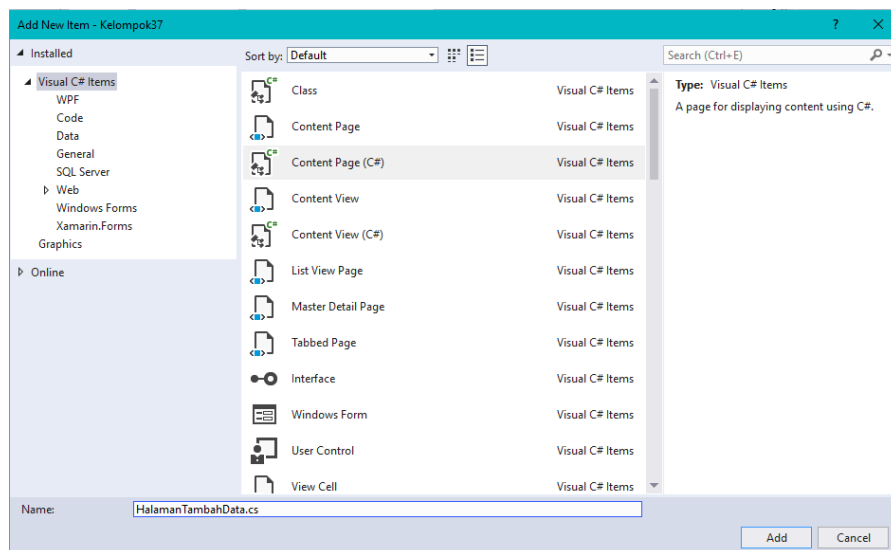
namespace Kelompok37.Model
{
    public class DataMahasiswa
    {
        [PrimaryKey]
        public int Id { get; set; }
        public string Nama { get; set; }
        public string Jurusan { get; set; }

        public override string ToString()
        {
            return "Nama : " + this.Nama + " ==> Jurusan : "
+ this.Jurusan;
        }
    }
}
```

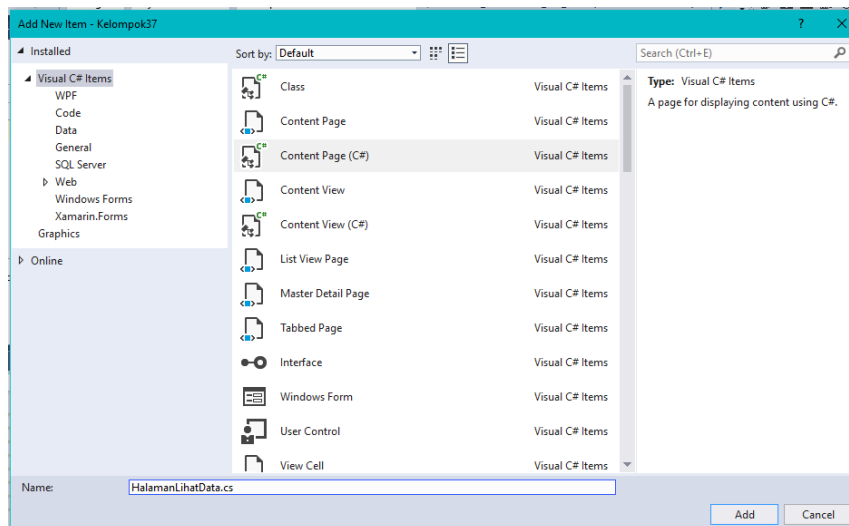
5. Membuat *file* Content Page (C#) pada folder View, masing – masing *file* diberi nama HalamanUtama.cs, HalamanTambahData.cs dan HalamanLihatData.cs. Klik kanan pada folder View> Add > New item > Content Page (C#)



Gambar 2.26 Membuat Content Page (C#) pada folder View dengan nama HalamanUtama.cs



Gambar 2.27 Content Page (C#) pada folder View dengan nama HalamanTambahData.cs



Gambar 2.28 Membuat Content Page (C#) pada folder View dengan nama HalamanLihatData.cs

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

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Xamarin.Forms;

namespace Kelompok37.View
{
    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());
    }
}
}
}

```

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

```

using Kelompok37.Model;
using SQLite;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

using Xamarin.Forms;

namespace Kelompok37.View
{
    public class HalamanTambahData : ContentPage
    {
        private Entry _nama;
        private Entry _jurusan;
        private Button _simpan;

        string _dbPath =
Path.Combine(System.Environment.GetFolderPath(System.Environment.
nt.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 += _simpan_Clicked;
            stackLayout.Children.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();
    }
}

```

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

```

using Kelompok37.Model;
using SQLite;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

using Xamarin.Forms;

namespace Kelompok37.View
{
    public class HalamanLihatData : ContentPage
    {
        private ListView _listView;
        string _dbPath =
Path.Combine(System.Environment.GetFolderPath(System.Environment.
nt.SpecialFolder.Personal), "myDB.db4");

        public HalamanLihatData()
        {
            this.Title = "Data Mahasiswa";

            var db = new SQLiteConnection(_dbPath);

```



```

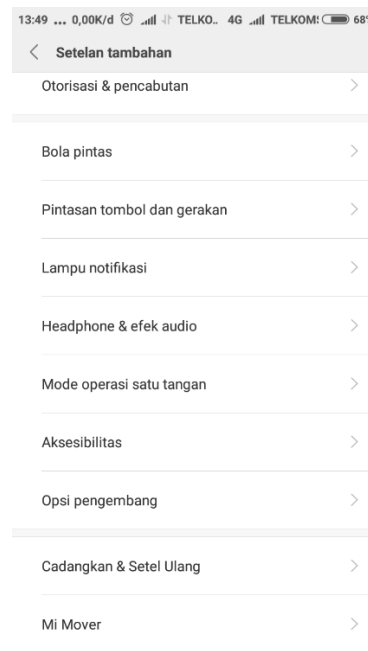
        StackLayout stackLayout = new StackLayout();

        _listView = new ListView();
        _listView.ItemsSource =
db.Table<DataMahasiswa>().OrderBy(x => x.Nama).ToList();
        stackLayout.Children.Add(_listView);

        Content = stackLayout;
    }
}
}

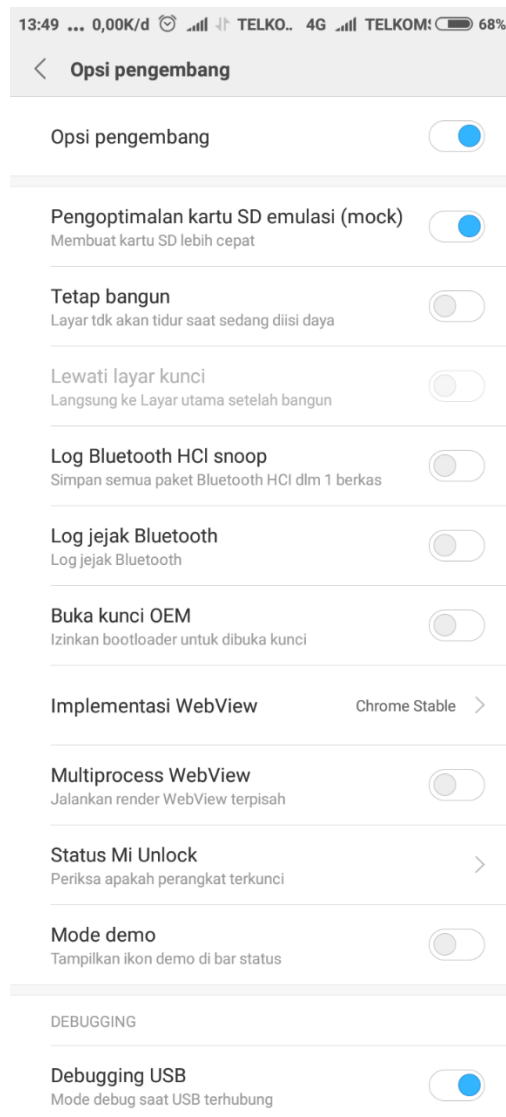
```

9. Penjelasan\_\_ : 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**.
10. Sebelum melakukan Build Project dan *Deploy* Project pastikan *smartphone* sudah terdapat menu Developer options.



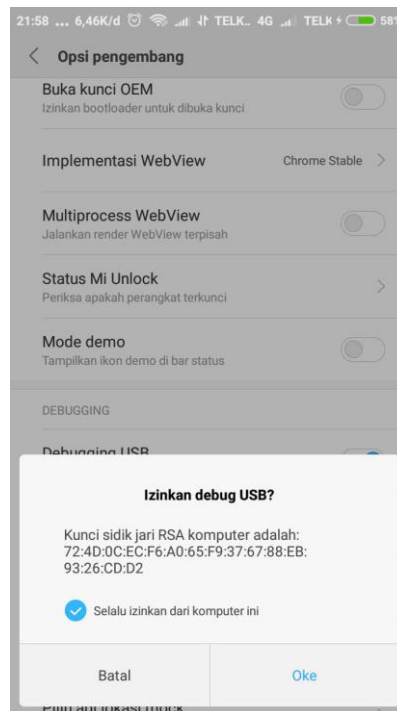
Gambar 2.29 Memilih Opsi Pengembang pada Android Phone

11. Pada menu Developer options checklist pada checkbox USB debugging



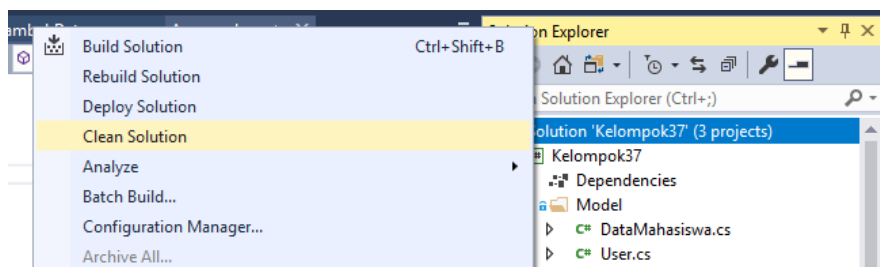
Gambar 2.30 Mengaktifkan Debugging USB

12. Ketika *smartphone* sudah terhubung melalui kabel USB dengan PC atau Laptop maka akan muncul dialog seperti berikut. Pilih OK



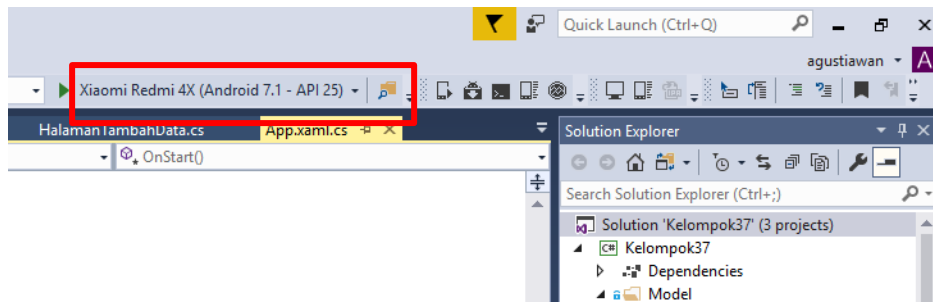
Gambar 2.31 Mengaktifkan Debugging USB

13. Kemudian sebelum membuild project maka lakukan Clean Solution terlebih dahulu pada Project Solution Kelompok37.



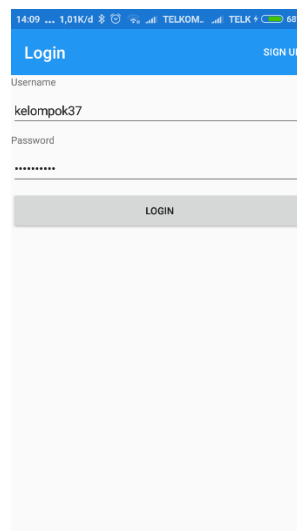
Gambar 2.32 Clean Solution

14. Apabila *smartphone* sudah terhubung dengan PC atau Laptop maka akan terdeteksi seperti pada gambar dibawah ini. Untuk melakukan build project sekaligus deploy project dapat dilakukan dengan langsung menekan tombol play.



Gambar 2.33 Tampilan Pada Visual Studio 2017 ketika *smartphone* telah terhubung

15. Apabila aplikasi sudah sukses dibangun dan *dideploy* maka akan muncul tampilan seperti pada gambar dibawah ini.



Gambar 2.34 Tampilan awal pada *smartphone* ketika program di *deploy*

## 2.5. Analisa Hasil Percobaan

### 2.5.1. Percobaan 1

Pada percobaan 1, praktikan membuat login pada aplikasi Xamarin. Sebelum program di *deploy* ke *smartphone*, *smartphone* sudah harus dalam keadaan USB Debugging agar program dapat diinstall menggunakan kabel USB .

Sebelum melakukan *deploy* program, dilakukan *Clean Project* terlebih dahulu. Setelah program di *deploy* pada *smartphone*, maka aplikasi akan otomatis terbuka dan menampilkan halaman *login*. Pada halaman ini user diminta untuk memasukkan data username dan password. Namun jika belum mempunyai akun maka user dapat membuat akun terlebih dahulu pada *button* Sign Up di samping kanan atas layar *smartphone*. Di halaman Sign Up, *user* diminta memasukkan *username*, *password*, dan alamat *email*.

Setelah *login* berhasil maka akan ditampilkan halaman utama yang berisi *button* Tambah Data dan Lihat Data. *Button* Tambah Data digunakan untuk menambahkan data mahasiswa berupa nama dan jurusan. Sedangkan *button* Lihat Data digunakan untuk melihat data yang sudah dimasukkan.

Berikut ini adalah *sourcecode* percobaan *login* :

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Kelompok37.Model
{
    public class User
    {
        public string Username { get; set; }

        public string Password { get; set; }

        public string Email { get; set; }
    }
}
```

Dapat dilihat pada *sourcecode* diatas bahwa *user* memiliki beberapa parameter atau data yaitu *Username*, *Password*, dan *Email*.

```
namespace Kelompok37
{
    public static class UserLogin
    {
        public static string Username = "Kelompok37";
        public static string Password = "password";
    }
}
```

```
}
}
```

*Sourcecode* diatas digunakan untuk *login*. Untuk masuk ke halaman utama, maka user dapat memasukkan username berupa “Kelompok37” dan mengisi password yaitu “password”.

```
<?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="Kelompok37.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>
```

*Sourcecode* diatas merupakan isi dari *file* LoginPage.xaml. XAML diatas digunakan untuk menentukan konten visual dari halaman *Login*. XAML sendiri menentukan konten visual suatu halaman dan bekerja sama dengan *file* C# di belakang kode.

Ketika user melakukan klik pada *button* Login maka perintah tersebut akan diproses pada LoginPage.xaml.cs. Berikut adalah *sourcecode* LoginPage.xaml.cs :

```
using Kelompok37.Model;
using Kelompok37.View;
using System;
using Xamarin.Forms;

namespace Kelompok37
{
    public partial class LoginPage : ContentPage
    {
        public LoginPage()
        {
            InitializeComponent();
        }
    }
}
```

```

e)      async void OnSignUpButtonClicked(object sender, EventArgs
        {
            await Navigation.PushAsync(new SignUpPage());
        }

e)      async void OnLoginButtonClicked(object sender, EventArgs
        {
            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;
        }
    }
}

```

*Sourcecode* diatas digunakan untuk menentukan apakah username dan password yang dimasukkan user benar atau salah. Jika username dan atau password yang dimasukkan user salah maka akan muncul messageLabel “Login Failed” dibawah *button* Login. Namun apabila *username* dan password yang dimasukkan user benar, maka akan ditampilkan Halaman Utama.

Sebelumnya, jika user ingin membuat akun baru dapat membuat akun melalui *Button* Sign Up yang terdapat pada pojok kanan atas layar *smartphone*. Berikut adalah *sourcecode* SignUp.xaml yang digunakan untuk menentukan konten visual pada halaman Sign Up.

```

<?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="Kelompok37.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 halaman Sign Up, user dapat memasukkan data *username*, *password*, dan alamat email. Pada bagian bawah juga terdapat *button* Sign Up untuk menuju ke proses selanjutnya. Proses back-end halaman Sign Up adalah sebagai berikut sebagaimana *sourcecode* pada *SignUpPage.xaml.cs* :

```

using Kelompok37.Model;
using System;
using System.Linq;
using Xamarin.Forms;

namespace Kelompok37
{
    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.IsNullOrEmpty(user.Username) &&
!string.IsNullOrEmpty(user.Password) &&
!string.IsNullOrEmpty(user.Email) &&
user.Email.Contains("@"));
}
}

```

Setelah memasukkan *username*, *password*, dan alamat email dan user melakukan klik *button* Sign Up maka akan dilakukan pengecekan menggunakan pengkondisian, jika terdapat salah satu parameter yang belum diisi maka akan muncul message label “Sign Up Failed” namun tetap berada pada halaman Sign Up. Namun jika data sudah dimasukkan semua maka proses akan menuju halaman berikutnya yaitu Main page.

```

<?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="Kelompok37.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>

```

*Sourcecode* diatas adalah *sourcecode* MainPage.xaml yang digunakan untuk mengatur tampilan ketika user telah berhasil melakukan sign up. *File* xaml ini diikuti oleh MainPage.xaml.cs sebagai berikut :

```

using Kelompok37;
using System;
using Xamarin.Forms;

namespace Kelompok37
{
    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();
        }
    }
}

```

*File* diatas digunakan untuk melakukan kondisi pada halaman main page. Halaman main page sendiri akan muncul ketika user telah melakukan sign up. Pada halaman main page, user hanya bisa melakukan log out dan ketika user telah melakukan log out maka user harus melakukan proses login lagi. Karena melakukan proses login lagi, maka masuk ke halaman LoginPageCS.cs. Berikut adalah *listing code* LoginPageCS.cs :

```

using Kelompok37.Model;
using System;
using Xamarin.Forms;

namespace Kelompok37
{
    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
        }
    };
}

e) async void OnSignUpButtonClicked(object sender, EventArgs

{
    await Navigation.PushAsync(new SignUpPageCS());
}

e) async void OnLoginButtonClicked(object sender, EventArgs

{
    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;
        }
    }
}

```

*Sourcecode* diatas berfungsi untuk mengatur fungsi-fungsi dan tampilan pada halaman *login*. Berdasarkan source diatas, yang dilakukan pertama kali yaitu mengatur bentuk *layout* halaman *login*. Halaman ini diberi judul Login, terdapat label untuk memasukkan *username* dan password, serta juga terdapat *button* dengan tulisan Login. Kemudian atur fungsi-fungsi yang dibutuhkan untuk menjalankan halaman *login*. User diminta untuk memasukkan username dan password yang sudah dibuat sebelumnya. Jika *username* dan password yang dimasukkan benar, maka user akan diarahkan ke *main page*. Tetapi apabila data yang dimasukkan salah, maka akan muncul pesan “Login Failed” dan diminta untuk memasukkan username dan password lagi.

```

using Kelompok37;
using Kelompok37.Model;
using System;
using System.Linq;
using Xamarin.Forms;

namespace Kelompok37
{
    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
    }
};
}

e) async void OnSignUpButtonClicked(object sender, EventArgs
{
    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.IsNullOrEmpty(user.Username) &&
!string.IsNullOrEmpty(user.Password) &&
!string.IsNullOrEmpty(user.Email) &&
user.Email.Contains("@"));
}

```

```
}
}
```

Source code tersebut merupakan *sourcecode* pada SignUpCS.cs yang berfungsi untuk mengatur tampilan halaman sign up dan juga mengatur fungsi-fungsi yang diinginkan. Halaman ini berjudul Sign Up, terdapat bagian dimana user dapat memasukkan nama, password, dan email. Dibawah bagian untuk menulis data tersebut terdapat *button* Sign Up. Jika data yang diisi sesuai maka akan beralih ke halaman main page. Namun jika gagal maka akan muncul label “Sign Up failed”.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Kelompok37.View;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

[assembly: XamlCompilation(XamlCompilationOptions.Compile)]
namespace Kelompok37
{
    public partial class App : Application
    {
        public static bool IsUserLoggedIn { get; internal set; }
        public App()
        {
            if (!IsUserLoggedIn)
            {
                MainPage = new NavigationPage(new LoginPage());
            }
            else
            {
                MainPage = new NavigationPage(new HalamanUtama());
            }
        }

        protected override void OnStart()
        {
            // Handle when your app starts
        }

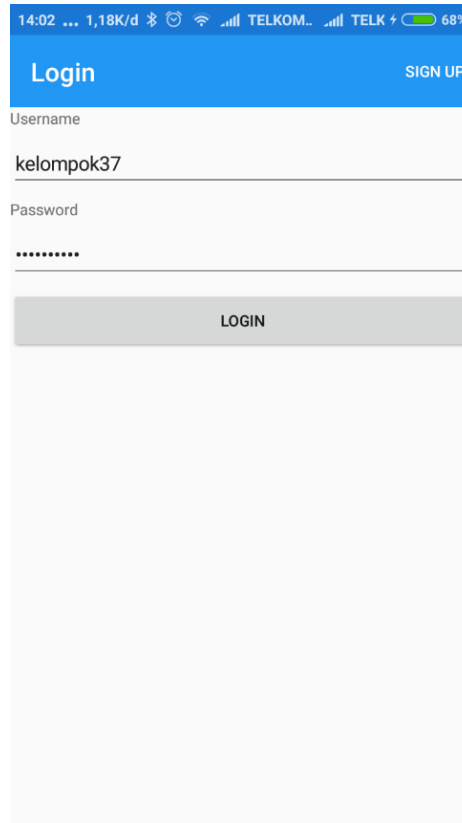
        protected override void OnSleep()
        {
            // Handle when your app sleeps
        }

        protected override void OnResume()
        {
            // Handle when your app resumes
        }
    }
}
```

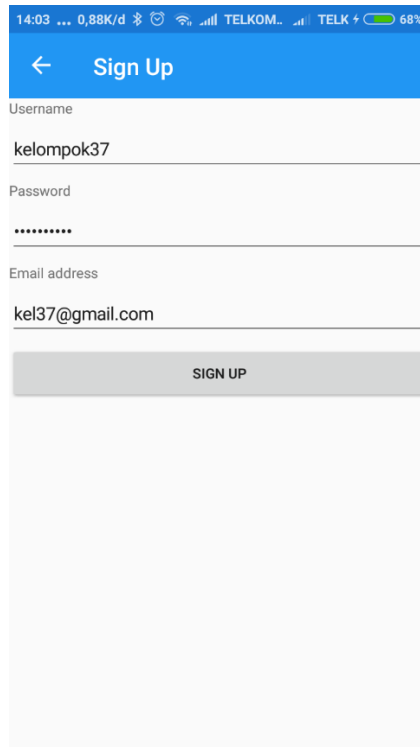
```
}  
    }  
}
```

*Sourcecode* diatas merupakan *sourcecode* App.xaml.cs yang digunakan untuk mengatur class yang akan dijalankan pertama kali saat aplikasi dijalankan.

Berikut adalah screenchot percobaan 1 :

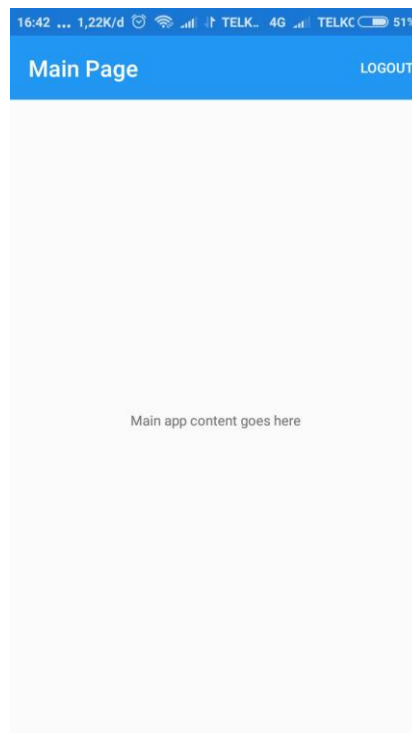


Gambar 2.35 Halaman Login



The screenshot shows a mobile application interface for a 'Sign Up' page. At the top, there is a blue header bar with a back arrow icon and the text 'Sign Up'. Below the header, the page has a light gray background. There are three input fields: 'Username' with the text 'kelompok37', 'Password' with masked characters '\*\*\*\*\*', and 'Email address' with the text 'kel37@gmail.com'. Below these fields is a gray button labeled 'SIGN UP'. The status bar at the top shows the time as 14:03, data usage as 0,88K/d, and battery level as 68%.

Gambar 2.36 Halaman SignUp



Gambar 2.37 Halaman Main Page



### 2.5.2 Percobaan 2

Pada percobaan 2, dilakukan pembuatan aplikasi Xamarin dengan memanfaatkan *database* SQLite. SQLite didownload dan diinstall terlebih dahulu sebelumnya.

Berikut adalah *sourcecode* class DataMahasiswa yang dibuat pada folder model. Pada class ini terdapat beberapa parameter pada class ini antara lain ID, Nama, dan Jurusan.

```
using SQLite;
using System;
using System.Collections.Generic;
using System.Text;

namespace Kelompok37.Model
{
    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;
        }
    }
}
```

Kemudian terdapat class HalamanUtama.cs yang digunakan untuk membuat halaman utama dengan judul “Data Mahasiswa”. Pada class ini terdapat pengaturan untuk mengatur layout tampilan dan juga terdapat *button* Tambah Data dan Lihat Data. Apabila user menekan *button* Tambah Data maka user akan diarahkan ke halaman tambah data dan menambahkan data yang diinginkan. Apabila menekan tombol Lihat Data maka user akan diarahkan ke halaman lihat data dan dapat melihat data apa saja yang ada pada halaman lihat data. Berikut adalah *sourcecode* HalamanUtama.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Xamarin.Forms;
```

```

namespace Kelompok37.View
{
    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());
        }
    }
}

```

Ketika user melakukan klik pada *button* Tambah Data, maka muncul tampilan untuk menambahkan data. Pengaturan tampilan dan fungsi pada Halaman ini diatur oleh HalamanTambahData.cs. Halaman ini berjudul Tambah Data Mahasiswa. Terdapat bagian untuk memasukkan nama dan jurusan serta terdapat *button* tambah data untuk menyimpan data yang telah dimasukkan tadi. Data yang disimpan akan disimpan di *database* SQLite yang telah diinstall sebelumnya. Apabila user melakukan klik Tambah Data maka akan muncul message atau pesan “Data xxx berhasil disimpan” dimana xxx adalah nama yang tadi dimasukkan. Setelah itu akan kembali ke Halaman Utama. Berikut adalah *sourcecode* HalamanTambahData.cs

```

using Kelompok37.Model;
using SQLite;

```

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

using Xamarin.Forms;

namespace Kelompok37.View
{
    public class HalamanTambahData : ContentPage
    {
        private Entry _nama;
        private Entry _jurusan;
        private Button _simpan;

        string _dbPath =
Path.Combine(System.Environment.GetFolderPath(System.Environment.S
pecialFolder.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 += _simpan_Clicked;
            stackLayout.Children.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();
    }
}

```

Setelah kembali ke halaman utama, maka untuk melihat data yang telah dimasukkan yaitu dengan menekan *button* Lihat Data pada halaman utama. Untuk melihat data menggunakan HalamanLihatData.cs. Halaman inilah yang mengatur layout dan fungsi-fungsi yang ada. Halaman ini berjudul Data Mahasiswa dan menampilkan data-data mahasiswa yang telah disimpan pada SQLite. Berikut adalah *sourcecode* HalamanLihatData.cs,

```

using Kelompok37.Model;
using SQLite;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

using Xamarin.Forms;

namespace Kelompok37.View
{
    public class HalamanLihatData : ContentPage
    {
        private ListView _listView;
        string _dbPath =
Path.Combine(System.Environment.GetFolderPath(System.Environment.S
pecialFolder.Personal), "myDB.db4");

        public HalamanLihatData()
        {
            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;
    }
}

```

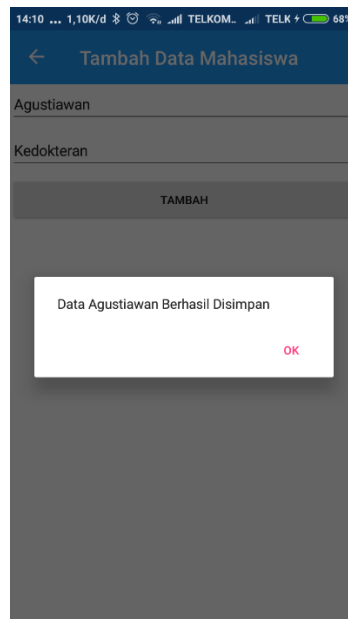
Pada HalamanUtama.cs, HalamanTambahdata.cs, dan HalamanLihatData.cs terdapat variabel `_dbpath` yang mana merupakan *database* lokal dengan nama `myDB.db4`. Berikut adalah *screenshot* hasil percobaan 2,



Gambar 2.38 HalamanUtama



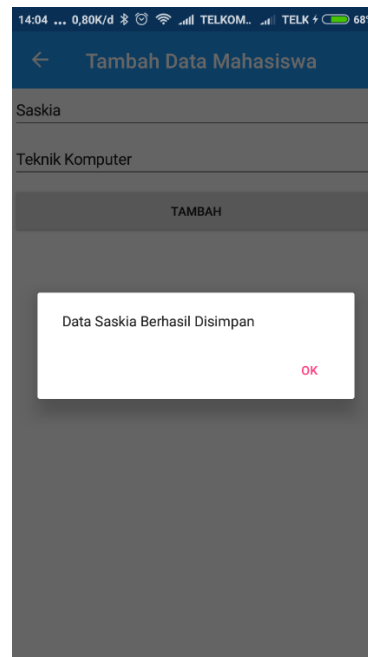
Gambar 2.39 Menambah data Agustiawan



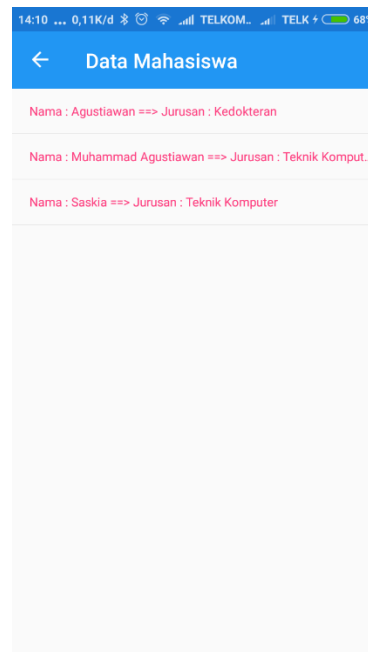
Gambar 2.40 Tambah data Agustiawan berhasil



Gambar 2.41 Menambah data Saskia



Gambar 2.42 Tambah data saskia berhasil



Gambar 2.43 Halaman Lihat Data

## 2.6. Tugas

Tugas pada bab ini, yaitu praktikan diminta untuk menambahkan menu edit data dan menu hapus data. Untuk itu dibuatlah beberapa class baru pada folder view, yaitu `HalamanEditData.cs` dan `HalamanHapusData.cs`. Selain itu ada penambahan code pada Halaman Utama.

Pada `HalamanUtama.cs` ditambahkan beberapa kodingan untuk membuat *button* Edit Data dan Hapus Data, *sourcecode* `HalamanUtama.cs` yang baru adalah sebagai berikut,

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Xamarin.Forms;

namespace Kelompok37.View
{
    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);

            button = new Button();
            button.Text = "Edit Data";
            button.Clicked += Button_Edit_Clicked;
            stacklayout.Children.Add(button);

            button = new Button();
            button.Text = "Hapus Data";
            button.Clicked += Button_Hapus_Clicked;
            stacklayout.Children.Add(button);

            Content = stacklayout;
        }
    }
}
```



```

        private async void Button_Hapus_Clicked(object sender,
EventArgs e)
        {
            await Navigation.PushAsync(new HalamanHapusData());
        }

        private async void Button_Edit_Clicked(object sender,
EventArgs e)
        {
            await Navigation.PushAsync(new HalamanEditData());
        }

        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());
        }
    }
}

```

Dalam menambah *button* edit, yang dilakukan yaitu menambahkan *sourcecode* `button = new button ();` kemudian *button* tersebut diberi tulisan yang akan ditampilkan di `button.Text = "Edit Data"` untuk *button* edit. Kemudian ditambahkan `button.Clicked += Button_Edit_Clicked;` dimana ketika *button* diklik nanti akan mengarahkan ke content di folder view yang bernama `HalamanEditData.cs`. Untuk *button* hapus, yang dilakukan yaitu menambahkan *sourcecode* `button = new button ();` kemudian *button* tersebut diberi tulisan yang akan ditampilkan di `button.Text = "Hapus Data"` untuk *button* edit. Kemudian ditambahkan `button.Clicked += Button_Hapus_Clicked;` dimana ketika *button* diklik nanti akan mengarahkan ke content di folder view yang bernama `HalamanHapusData.cs`

Kemudian membuat `HalamanEditData.cs` di folder view, berikut adalah *sourcecode* `HalamanEditData.cs`

```

using Kelompok37.Model;
using Kelompok37.View;
using SQLite;

```

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

using Xamarin.Forms;

namespace Kelompok37.View
{
    public class HalamanEditData : ContentPage
    {
        private ListView _listView;
        private Entry _id;
        private Entry _nama;
        private Entry _jurusan;
        private Button _ubah;

        DataMahasiswa dataMahasiswa = new DataMahasiswa();

        string _dbPath =
Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), "myDB.db4");

        public object Popasync { get; private set; }

        public HalamanEditData()
        {
            this.Title = "Edit Data";

            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);

            _id = new Entry();
            _id.Placeholder = "Id";
            _id.IsVisible = false;
            stackLayout.Children.Add(_id);

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

            _jurusan = new Entry();
            _jurusan.Keyboard = Keyboard.Text;
            _jurusan.Placeholder = "Jurusan";
            stackLayout.Children.Add(_jurusan);

```

```

        _ubah = new Button();
        _ubah.Text = "Ubah";
        _ubah.Clicked += _ubah_Clicked;
        stackLayout.Children.Add(_ubah);

        Content = stackLayout;
    }

    private async void _ubah_Clicked(object sender, EventArgs
e)
    {
        var db = new SQLiteConnection(_dbPath);
        DataMahasiswa dataMahasiswa = new DataMahasiswa()
        {
            Id = Convert.ToInt32(_id.Text),
            Nama = _nama.Text,
            Jurusan = _jurusan.Text
        };
        db.Update(dataMahasiswa);
        await Navigation.PopAsync();
    }

    private void _listView_ItemSelected(object sender,
SelectedItemChangedEventArgs e)
    {
        dataMahasiswa = (DataMahasiswa)e.SelectedItem;
        _id.Text = dataMahasiswa.Id.ToString();
        _nama.Text = dataMahasiswa.Nama;
        _jurusan.Text = dataMahasiswa.Jurusan;
    }
}

```

Pada HalamanEditData.cs dilakukan inisialisasi Listview dengan variabel `_id`, `_nama`, `_jurusan`, `_ubah`, yang nantinya digunakan untuk menyimpan data yang diubah pada halama edit ini. Selanjutnya yaitu membuat kelas baru bernama mahasiswa. Disini kita juga tambahkan *databasenya* sama seperti di halaman utama. Lalu kita buat judul pada halaman edit ini dengan nama “Edit Data”. Lalu kita tambahkan Entry `_id`, `_nama`, dan jurusan untuk ketika kita membuka halaman tersebut, lalu kita mengeklik data yang ada akan muncul entry untuk mengganti nama dan jurusan. Lalu kita buat *button* `_ubah` yang nantinya ketika kita sudah selesai mengedit kita tekan simpan data akan langsung di ganti di *database*.

Selanjutnya dibuat HalamanHapusData.cs yang digunakan untuk menghapus data mahasiswa. Berikut adalah *sourcecode* HalamanHapusData.cs.

```
using Kelompok37.Model;
using SQLite;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

using Xamarin.Forms;

namespace Kelompok37.View
{
    public class HalamanHapusData : ContentPage
    {
        private ListView _listView;
        private Button _hapus;

        DataMahasiswa dataMahasiswa = new DataMahasiswa();

        string _dbPath =
Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), "myDB.db4");

        public HalamanHapusData()
        {
            this.Title = "Hapus Data";

            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);

            _hapus = new Button();
            _hapus.Text = "Hapus";
            _hapus.Clicked += _hapus_Clicked;
            stackLayout.Children.Add(_hapus);

            Content = stackLayout;
        }

        private async void _hapus_Clicked(object sender, EventArgs e)
        {
            var db = new SQLiteConnection(_dbPath);
            db.Table<DataMahasiswa>().Delete(x => x.Nama ==
dataMahasiswa.Nama);
            await Navigation.PopAsync();
        }
    }
}
```

```

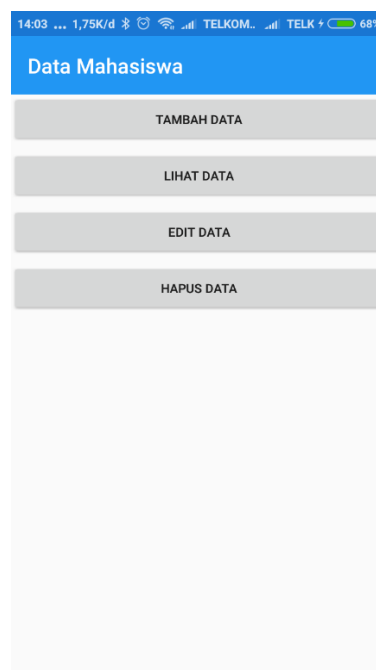
    }

    private void _listView_ItemSelected(object sender,
SelectedItemChangedEventArgs e)
    {
        dataMahasiswa = (DataMahasiswa)e.SelectedItem;
    }
}

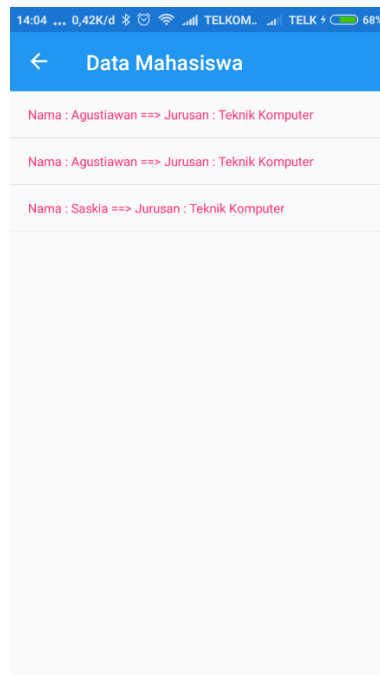
```

Pada HalamanHapusData.cs ini pertama-tama yaitu menginisialisasi ListView dan juga *button* \_hapus. Listview digunakan untuk menampilkan data yang ada di datamahasiswa. Ketika masuk ke halaman hapus data, maka akan ditampilkan data mahasiswa yang ada. Kemudian untuk menghapus data mahasiswa dengan cara meng-klik data yang ada pada daftar dan kemudian klik tombol hapus.

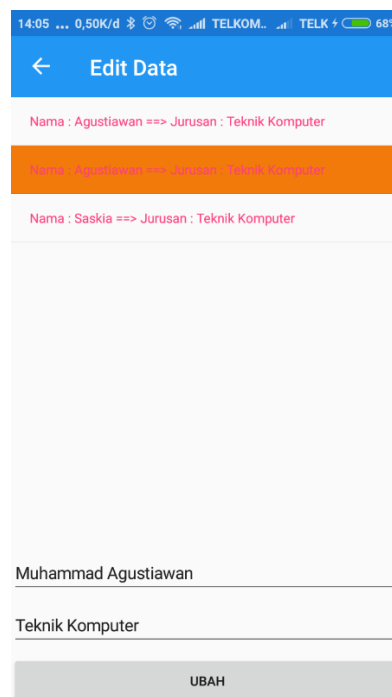
Berikut adalah hasil *screenshot* tugas ini,



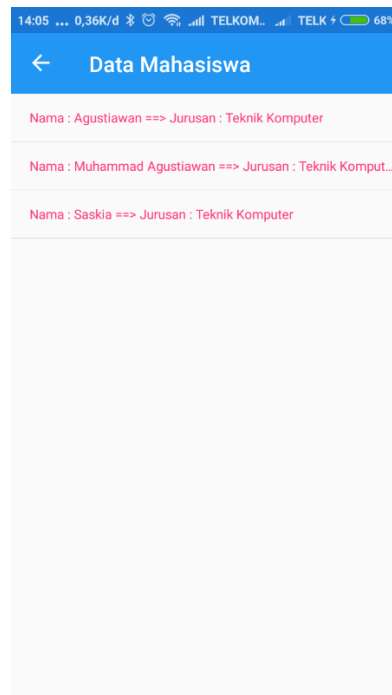
Gambar 2.44 Tampilan Halaman Utama



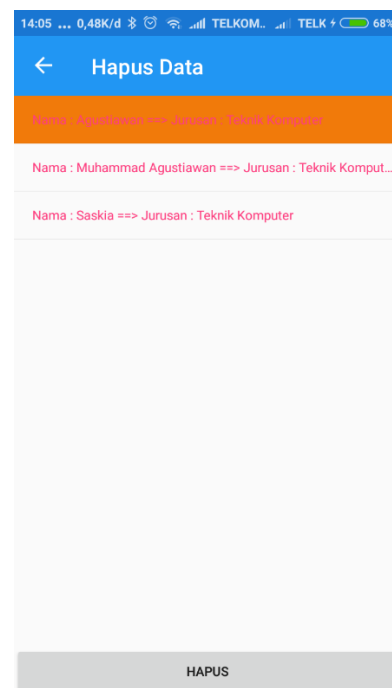
Gambar 2.45 Tampilan Awal Lihat Data



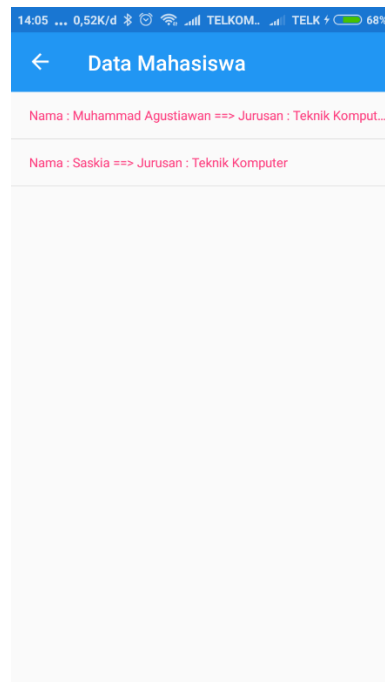
Gambar 2.46 Mengubah Data Agustiawan



Gambar 2.47 Data Agustiawan berubah menjadi Muhammad Agustiawan



Gambar 2.48 Menghapus Data Agustiawan



Gambar 2.49 Data Agustiawan telah terhapus

(Link github : [github.com/agustiawan0798/appmodul1kelompok37](https://github.com/agustiawan0798/appmodul1kelompok37))



## 2.7. Kesimpulan

1. *Database* yang digunakan pada praktikum ini yaitu SQLite
2. Sebelum melakukan praktikum, harus *terinstall* terlebih dahulu SDK yang mana sesuai dengan perangkat android yang akan dipakai.
3. Untuk menggunakan *database* SQLite maka harus *diinstall* terlebih dahulu SQLite-net-pcl nya terlebih dahulu melalui menu Manage NuGet Package.
4. Untuk mengatur layout atau tampilan suatu halaman dapat menggunakan *file.xml*
5. Variabel *\_dbpath* digunakan untuk membuat suatu halaman dapat terhubung dengan *database* lokal dalam hal ini yaitu *database* dengan nam myDB.db4
6. Fungsi dari *file.cs* yaitu untuk mengatur fungsi-fungsi dan kondisi apa saja yang terdapat pada halaman tersebut.
7. Xamarin merupakan *platform* untuk pengembangan aplikasi *cross platform* di mana pengembangannya menggunakan bahasa C# dan .NET framework.