

PROJECT UAS
DATABASE KP B



UBAYA
UNIVERSITAS SURABAYA

Oleh:

Kelompok Lumba-Lumba

Joshua Nehemia Subagyo	160423034
Evan Daniel Tandiawan	160423033
Alvin Kurniawan	160423055
Fransiskus Hendra S. R	160423062
Vivian Sisca Maria	160423066

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS SURABAYA
2025

1. Cara Kerja Program	4
1.1 User Control Profile	4
1.2 User Control Profile Teman	6
1.3 User Control Daftar	7
1.4 User Control Daftar Konten	8
1.5 User Control Cari Teman.....	9
1.6 User Control Tambah Kisah Hidup.....	9
1.7 User Control Konten Detail	10
1.8 User Comtrol Login.....	11
1.9 UC Chat dan UC_ChatListItem.....	12
1.10 User Control Request Pertemanan	12
1.11 User Control Tambah Konten	14
1.12User Control Tambah Organisasi.....	14
1.13User Control Home.....	15
2. Susunan Class, Property, dan Method	16
2.1 Class Services	16
2.2 Class KoneksiDatabase.....	26
2.3 Class Repositories	29
2.4 Class StatusPertemanan	29
2.5 Class User	30
2.6 Class Kota	32
2.7 Class Organisasi	34
2.8 Class Teman	36
2.9 Class KisahHidup.....	37
2.11 Class Komen.....	42
2.12 Class DAO_Users.....	44
2.13 Class DAO_Kota.....	49
2.14 Class DAO_Organisasi.....	50

2.15 Class DAO_Teman.....	51
2.16 Class DAO_Komen	54
2.17 Class DAO_Konten.....	55
2.18 Class DAO_KisahHidup	57
2.19 Class DAO_Tag.....	58
3. Kreasi Fitur	58
4. Dokumentasi Kerja Kelompok	67
A. Dokumentasi Kerja Kelompok Pertama.....	67
B. Dokumentasi Kerja Kelompok Kedua.....	68
5. Dokumentasi Konsultasi Kakak Asdos	69
A. Konsultasi Pertama	69
B. Konsultasi Kedua.....	70

1. Cara Kerja Program

Aplikasi PamerYuk ini dirancang untuk mendukung berbagai interaksi pertemanan. Aplikasi ini memungkinkan pengguna untuk mendaftar, mencari teman berdasarkan organisasi, berkomunikasi melalui percakapan real-time, dan membuat serta berinteraksi dengan konten. Tujuan utama aplikasi ini adalah menciptakan platform yang memudahkan pengguna membangun koneksi sosial yang berbasis pada kesamaan pengalaman hidup.

Dalam pengembangannya, aplikasi ini kami rancang dengan tampilan yang user-friendly agar memudahkan pengguna di berbagai kalangan. Selain itu, efisiensi dan fleksibilitas juga menjadi prioritas utama kami. Kami menyediakan fitur tambahan, seperti sistem real-time chat dan kemampuan memberikan "like" pada konten, kami berharap dengan adanya fitur like ini, pengguna semakin *excited* dan tertarik. Dengan kombinasi fitur-fitur tersebut, aplikasi PamerYuk tidak hanya menjadi alat komunikasi, tetapi juga platform sosial yang mendukung hubungan, dan menciptakan ruang untuk berbagi cerita serta interaksi antar pengguna.

The screenshot shows the 'PROFIL AKUN' (User Profile) and 'KISAH HIDUP' (Life Stories) sections of the application. On the left, there are two placeholder boxes for profile pictures: 'Foto Diri:' (Self Photo) and 'Foto Profil:' (Profile Photo). The main area contains form fields for account information: 'Nama Lengkap:' (Full Name), 'No KTP:' (KTP Number), 'Username:' (Username), 'Tanggal Lahir:' (Birth Date), 'Email:' (Email), and 'Kota:' (City). Below these fields are a 'Simpan' (Save) button and a 'Tambah Kisah Hidup' (Add Life Story) button. To the right, a large vertical list box labeled 'listBoxKisahHidup' displays a list of life stories.

PROFIL AKUN		KISAH HIDUP
Foto Diri:		<input type="text" value="listBoxKisahHidup"/>
Foto Profil:		<input type="button" value="Tambah Kisah Hidup"/>
Form Fields:		
Nama Lengkap: <input type="text"/>		No KTP: <input type="text"/>
Username: <input type="text"/>		Tanggal Lahir: <input type="date" value="Saturday , 11 Jar"/>
Email: <input type="text"/>		Kota: <input type="text"/>
<input type="button" value="Simpan"/>		

Digunakan untuk menampilkan data pengguna yang sedang aktif. Pada UC ini, identitas dan data diri pengguna ditampilkan dengan lengkap. UC ini menyediakan opsi edit profil untuk mengubah data pengguna.

Kisah hidup ditampilkan menggunakan listbox, yang datanya berasal dari method `DisplayListKisahHidup()`. Setiap elemen di daftar kisah hidup diubah ke dalam bentuk string lalu dimasukkan ke dalam list `listBoxKisahHidup`.

Komponen Form Akun:

1) TextBox:

- TextBoxNamaLengkap: Menampilkan dan mengubah nama lengkap pengguna.
- textBoxUsername: Menampilkan dan mengubah username pengguna.
- textBoxEmail: Menampilkan dan mengubah email pengguna.
- textBoxNoKTP: Menampilkan nomor KTP pengguna.

2) Panel:

- panelFotoProfil: Menampilkan foto profil pengguna.
- panelFotoDiri: Menampilkan foto diri pengguna.

3) ListBox:

- listBoxKisahHidup: Menampilkan list kisah hidup yang sudah diinputkan pengguna melalui buttonTambahKisahHidup.

4) Button

- buttonSimpan: Menyimpan perubahan data profil pengguna.
- buttonTambahKisahHidup: Berpindah ke UC_TambahKisahHidup untuk menambahkan kisah hidup yang dimiliki pengguna.

5) DateTimePicker

- datePickerTglLahir: Menampilkan pilihan tanggal lahir pengguna.

6) Combo Box

- comboBoxKota: Menampilkan pilihan kota asal pengguna.

1.1 User Control Profile Teman



Berfungsi untuk menampilkan informasi profil teman yang dipilih pengguna dan menampilkan fitur daftar konten yang dimiliki oleh teman.

Komponen Form Akun Teman:

- 1) Label:
 - labelUsername
 - labelBoxKota
 - labelBoxTanggalLahir
- 2) FlowLayoutPanel:
 - flowLayoutPanelKonten
- 3) Panel:
 - panelFotoProfil

1.2 User Control Daftar

The screenshot shows a Windows application window titled "MainForm". On the left side, there is a sidebar with two items: "1. Register Username dan Password" and "2. Melengkapi data diri", both of which have checkboxes next to them. The main area is titled "Daftar Akun" and contains four input fields: "Username", "Password", "Confirm Password", and "Lanjut" (Next). Below the "Lanjut" button is a link "Sudah memiliki akun? [Masuk](#)".

The screenshot shows a web-based application with a sidebar on the left containing the same two items: "1. Register Username dan Password" and "2. Melengkapi data diri". The main area is titled "Daftar Akun" and includes several input fields: "Nama Lengkap", "Email", "No KTP", "Kota" (dropdown), "Tanggal Lahir" (dropdown), "Foto Diri" (with a placeholder image and "Upload Image" button), "Foto Profil" (with a placeholder image and "Upload Image" button), and a large blue "Daftarkan Akun" (Register Account) button. Below the "Daftarkan Akun" button is a link "Sudah memiliki akun? [Masuk](#)".

Digunakan untuk membantu pengguna dalam melakukan pendaftaran ke dalam sistem aplikasi. Informasi yang perlu diinputkan pengguna adalah nama username, password, konfirmasi password, nama lengkap, email, nomor KTP, kota tempat tinggal, tanggal lahir, foto diri, dan foto profil.

Komponen form Daftar:

1) TextBox:

- textBoxUsername
- textBoxPassword
- textBoxKonfirmasi (Password dan konfirmasi password harus memiliki nilai yang sama sebelum melanjutkan ke proses pendaftaran.)
- textBoxNamaLengkap
- textBoxEmail
- textBoxNoKTP

2) ComboBox:

- comboBoxKota

3) DateTimePicker

- dateTimeTglLahir

4) Panel:

- panelFotoDiri
- panelFotoProfil

5) Button:

- buttonDaftar
- buttonUploadImage
- buttonMasuk

1.3 User Control Daftar Konten

Tambah Konten

Konten Saya

Dapat menghubungkan dua fitur yaitu “Tambah Konten” dan “Konten Saya”. Form ini dapat menampilkan dua fitur tanpa harus membuka *window* baru. Dengan memanfaatkan form ini, aplikasi ini menjadi lebih efisien dan responsive.

1.4 User Control Cari Teman

Cari Teman dan Ajukan Pertemanan!

Username : Organisasi :

Menyediakan fitur untuk mencari teman berdasarkan informasi username dan organisasi. Username menggunakan textbox sedangkan organisasi menggunakan comboBox, agar mendapat data yang valid. Selain itu, menyediakan fitur add friend, namun hanya muncul ketika username dan organisasi mendapat data yang valid.

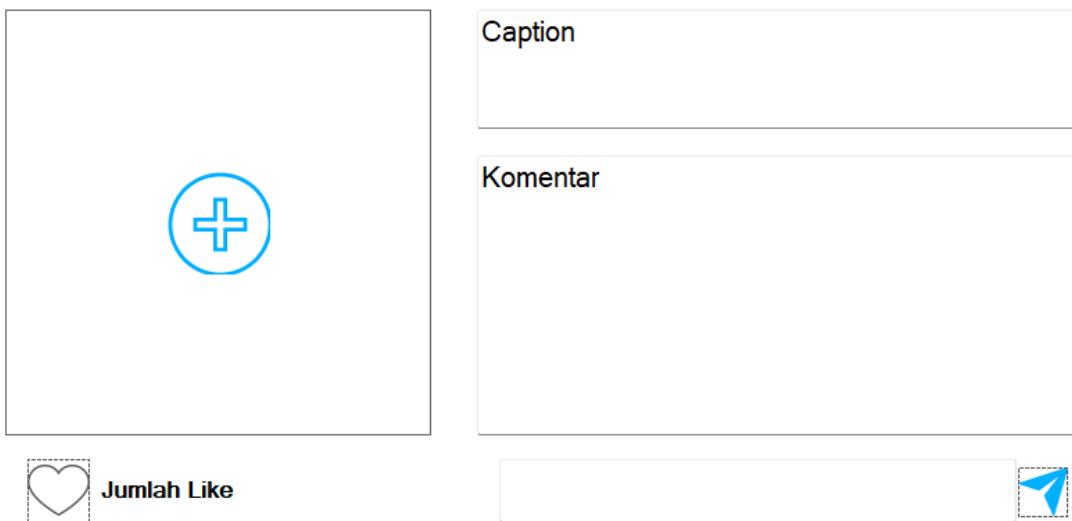
1.5 User Control Tambah Kisah Hidup

Kisah Hidup

Tahun Awal: Tahun Akhir: Organisasi:

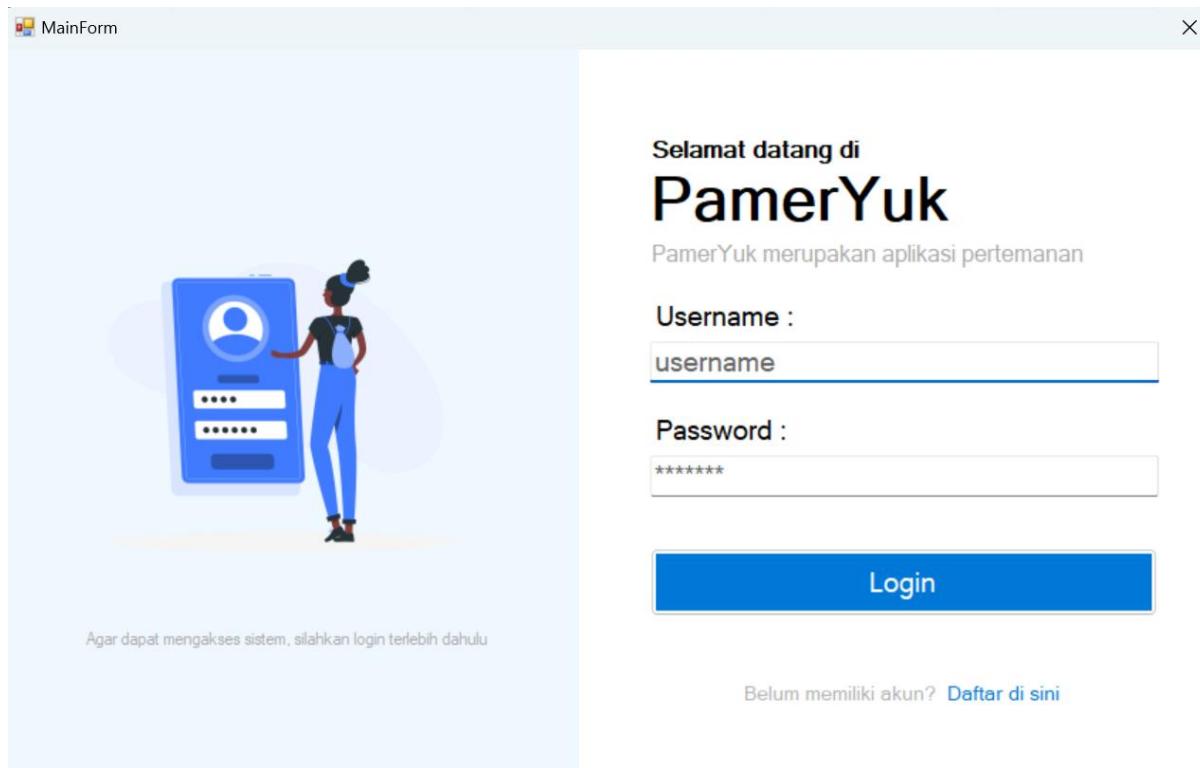
Menambahkan kisah hidup baru pengguna ke aplikasi, khususnya kisah hidup mereka yang berhubungan dengan organisasi. Organisasi menggunakan comboBox agar mendapat data yang valid. Tahun awal dan tahun akhir menggunakan numericUpDown, ini berguna untuk mencatat riwayat dan periode kisah. Deskripsi kisah hidup dapat ditulis dalam rich text box.

1.6 User Control Konten Detail



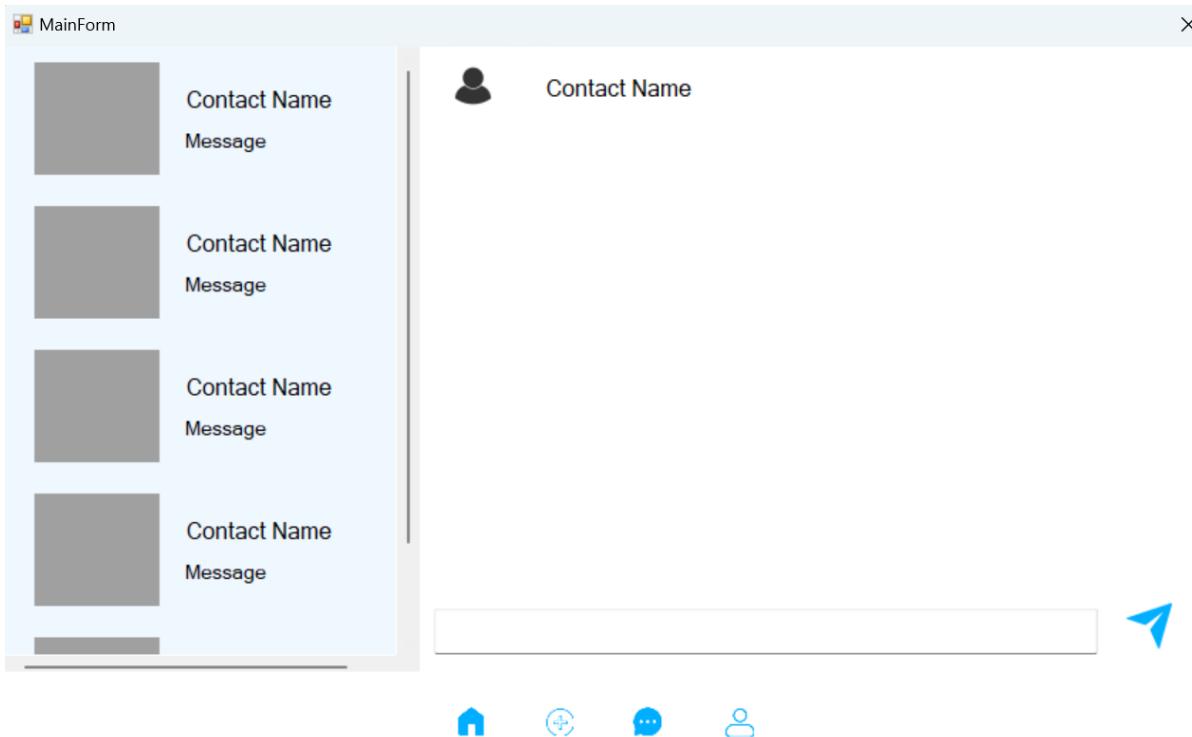
User Control ini berfungsi untuk menampilkan sebuah konten yang telah diupload oleh user. Pada user control ini, pengguna disajikan konten yang berisi foto/video, caption atau deskripsi konten, dan kolom komentar yang dapat menampilkan history komentar dari konten tersebut. Selain itu, apabila pengguna menyukai atau tertarik dengan konten yang disajikan, maka pengguna dapat memberikan like dan komentar untuk postingan tersebut.

1.7 User Control Login



Memastikan bahwa hanya username dan password yang terdaftar saja yang dapat mengakses aplikasi. Ketika username dan password valid dan pengguna memencet button Login, maka pengguna akan langsung diarahkan ke UC home. Namun, jika pengguna belum memiliki akun dan memencet button daftar, maka pengguna diarahkan ke UC daftarUsername.

1.8 UC Chat dan UC_ChatListItem



Memfasilitasi percakapan langsung kepada teman dalam aplikasi. Panel kiri memungkinkan pengguna melihat daftar kontak dan pesan terakhir mereka dengan mudah, membantu melacak percakapan. Panel kanan memungkinkan pengguna melakukan chat terpisah dengan kontak lain.



UC Chat memfasilitasi *user* untuk berkomunikasi melalui chat. Form ini menggunakan *FlowLayoutPanel* yang berisi UC_ChatListItem untuk menampilkan teman-teman *user* yang status pertemanannya adalah “Berteman” sesuai dengan *data*. *User* dapat memilih teman yang ingin *user* ajak berkomunikasi dengan mengirimkan pesan melalui *textbox* seperti aplikasi chatting pada umumnya.

1.9 User Control Request Pertemanan

A. List Undangan Pertemanan yang masuk

Username	Tanggal Kirim	Terima Permintaan Pertemanan	Tolak Permintaan Pertemanan
----------	---------------	--	---

Undangan pertemanan yang masuk memungkinkan user untuk mengkonfirmasi undangan pertemanan yang masuk. User dapat menerima atau menolak undangan pertemanan tersebut.

B. List Undangan pertemanan yang belum direspon

Username	Tanggal Kirim	Kirim Ulang Permintaan Pertemanan
----------	---------------	---

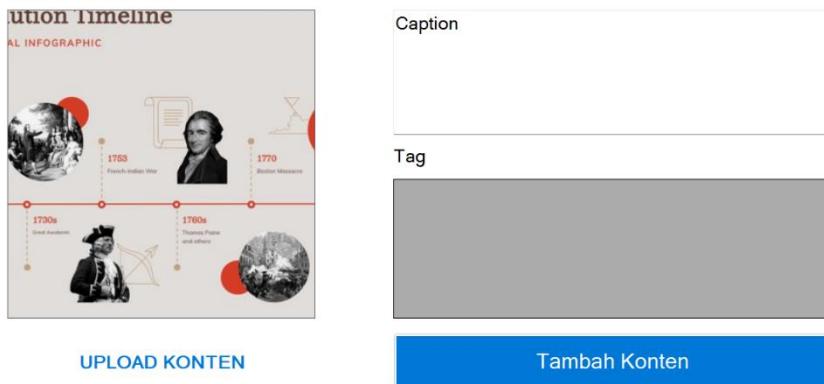
Sistem ini memungkinkan pengguna untuk mengirim undangan pertemanan. Namun, sangat memungkinkan bagi user yang menerima undangan tersebut tidak segera untuk menerima atau menolak undangan si pengirim. Oleh karena itu, sistem memungkinkan pengguna untuk mengirim kembali undangan pertemanan kepada orang yang sama.

C. Penggunaan *User Control Request* Pertemanan pada *section Request Pertemanan* di *UC_Home*

Request Pertemanan			
Masuk		Terkirim	
Username	Tanggal Kirim	Terima Permintaan Pertemanan	Tolak Permintaan Pertemanan

Gambar di atas merupakan implementasi dari kedua user control di atas. Pengguna dapat memilih salah satu dari dua opsi yang disediakan. Opsi pertama adalah “masuk” atau menampilkan semua undangan pertemanan yang masuk ke akun pengguna. Lalu, opsi kedua “Terkirim” adalah untuk menampilkan riwayat undangan yang sudah dikirim pengguna, namun belum direpon oleh penerima.

1.10 User Control Tambah Konten



User control ini memfasilitasi pengguna untuk dapat mengunggah konten berupa aktivitas maupun kesehariannya. Pada halaman ini, pengguna dapat mengunggah foto atau video yang ingin pengguna posting sekaligus memberikan *caption* untuk konten tersebut. Selain itu, pengguna dapat memention teman-teman dalam konten tersebut. *Button* tambah konten akan menyimpan konten tersebut ke *database*, dan dapat diakses oleh teman-teman pengguna melalui fitur detail konten.

1.11 User Control Tambah Organisasi

Tambah Organisasi

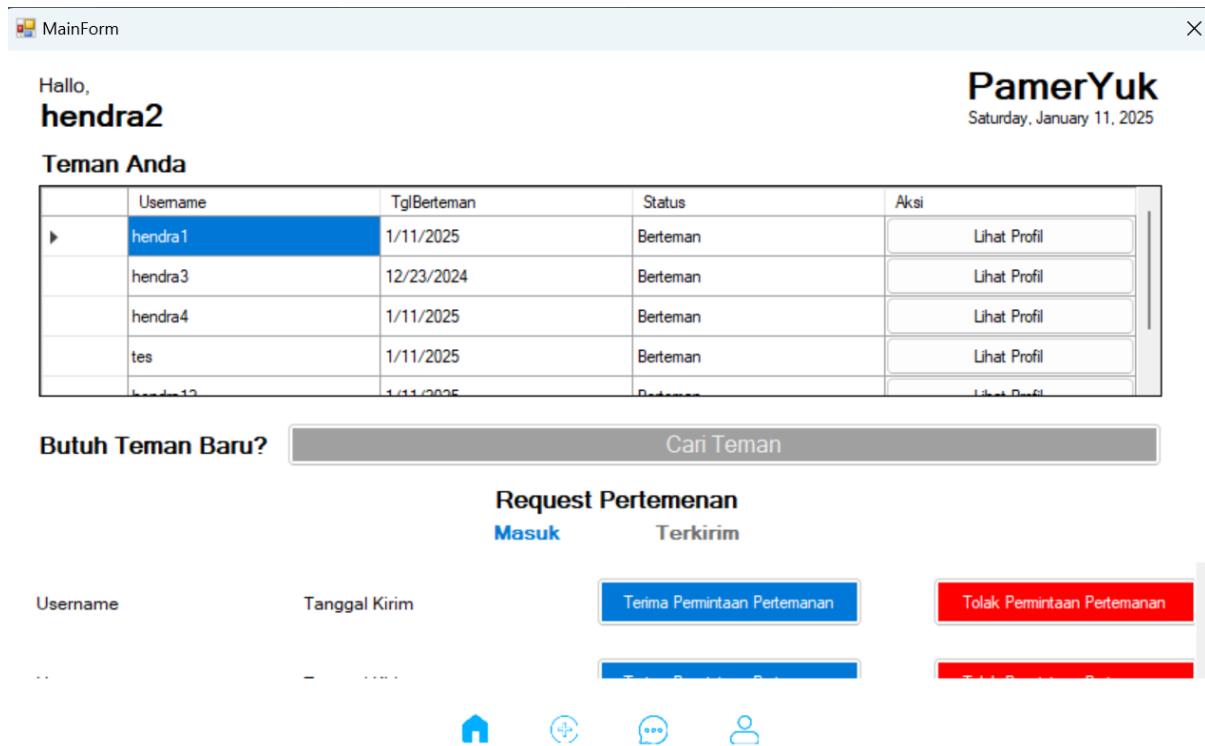
Organisasi:

Kota:

Simpan

Mempermudah pengguna untuk menambahkan informasi organisasi ke dalam *database*. Organisasi menggunakan *textbox* sedangkan kota menggunakan *comboBox*, hal ini agar data kota dapat tercatat secara valid. *Button* simpan akan menyimpan dan melakukan sinkronisasi pada *database*.

1.12 User Control Home



User Control Home merupakan Tampilan utama dari aplikasi PamerYuk. Halaman ini terbagi menjadi 4 section. Section pertama adalah Header, section kedua adalah List teman, section ketiga adalah Pencarian Teman, dan section terakhir adalah Request Pertemanan. Pada section Header, terdapat informasi berupa username pengguna, tanggal hari ini, dan nama aplikasi PamerYuk. Section List teman menampilkan daftar teman dari user. Selain itu, pada section ini pengguna dapat melihat detail profile dan konten teman melalui button "Lihat Profil". Selanjutnya, section pencarian teman memfasilitasi pengguna untuk dapat mencari teman baru melalui dengan cara klik button "Cari Teman". Section Request Pertemanan menampilkan kepada pengguna undangan Pertemanan yang masuk dan undangan pertemanan yang sudah terkirim namun belum ditanggapi.

2. Susunan *Class*, *Property*, dan *Method*

2.1 Class Services

```
1  using PamerYukLibrary;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7  using PamerYukLibrary;
8  using PamerYukLibrary.DAO;
9  using System.Diagnostics.Eventing.Reader;
10 using System.IO;
11 using System.Windows.Forms;
12 using System.Drawing;
13 using static System.Windows.Forms.VisualStyles.Window;
14 using PamerYukLibrary.Entity;
15
16 namespace PamerYukFormsApp
17 {
18     public class Service //On Load
19     {
20         #region VARIABLES
21         private User current_user;
22         private List<Kota> listKota;
23         private List<Organisasi> listOrganisasi;
24         private List<Teman> listTeman;
25         private string MediafilePath = @"C:\PamerYuk\";
26         private string MediafilePathDB = @"C:\\\\PamerYuk\\\";
27         public Konten buffer_konten;
28         #endregion
29     }
30 }
```

Bagian *References*:

1. using PamerYukLibrary;
- *Reference* ini digunakan untuk memanggil *library* yang telah kami buat. *Library* yang telah kami buat berisi *class Entity* dan *class DAO* yang memiliki fungsinya masing-masing.
2. using PamerYukLibrary.Entity;
- *Reference* ini digunakan untuk memanggil *class Entity* yang berada di *Library* yang telah kami buat. *Class* ini kami gunakan untuk melakukan proses enkapsulasi data-data dari *user*. Data-data pada *class Entity* nantinya akan diteruskan ke *class DAO* untuk diproses lebih lanjut.
3. using PamerYukLibrary.DAO;

- *Reference* ini digunakan untuk memanggil *class DAO* yang berada di *Library* yang telah kami buat. *Class* ini kami gunakan untuk mengubah data-data dari *user*, menjadi data-data yang dapat diterima dan diproses di *database*. *Method-method* yang ada di dalam *class* ini, digunakan untuk menjalankan *DML (Database Manipulation Language) commands* pada *database*.
- Bagian *Variables*:
 1. private User current_user;
 - *Variable* ini digunakan untuk menampung *user* yang sedang menggunakan sistem saat ini. *Variable* ini dibuat *private* karena nilainya akan diakses melalui *Property*.
 2. private List<Kota> listKota;
 - *Variable* ini digunakan untuk menampung daftar kota yang tersimpan di dalam *database*. *Variable* ini dibuat *private* karena nilainya akan diakses melalui *Property*.
 3. private List<Organisasi> listOrganisasi;
 - *Variable* ini digunakan untuk menampung daftar organisasi yang tersimpan di dalam *database*. *Variable* ini dibuat *private* karena nilainya akan diakses melalui *Property*.
 4. private List<Teman> listTeman;
 - *Variable* ini digunakan untuk menampung daftar teman dari *current_user*. *Variable* ini dibuat *private* karena nilainya akan diakses melalui *Property*.
 5. private string MediafilePath = @"C:\PamerYuk\";
 6. private string MediafilePathDB = @"C:\\PamerYuk\\\";
 7. public Konten buffer_konten;
 - *Variable* ini digunakan sebagai *variable* penampung *konten* yang ingin dibuat ataupun diubah.

```

#region CONSTRUCTOR
2 references
public Service()
{
    this.ListOrganisasi = DAO_Organisasi.Select_Organisasi();
    this.ListKota = DAO_Kota.Select_ListKota();
    CreateDirectory();
}
#endregion

#region PROPERTIES

33 references
public User Current_user { get => current_user; set => current_user = value; }
4 references
public List<Kota> ListKota { get => listKota; set => listKota = value; }
3 references
public List<Organisasi> ListOrganisasi { get => listOrganisasi; set => listOrganisasi = value; }
3 references
public List<Teman> ListTeman { get => listTeman; set => listTeman = value; }
#endregion

```

- Bagian *Constructor*:
 1. public Service()
 - *Constructor* ini dipanggil untuk mengambil data terkait daftar kota dan daftar organisasi yang ada di *database*. Tidak hanya itu, akan dibuat sebuah *directory* untuk menyimpan data
- Bagian *Property* :
 1. public User Current_user
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable current_user*, yaitu *user* yang sedang menggunakan sistem saat ini.
 2. public List<Kota> ListKota
 - *Property* ini digunakan untuk mengakses / mengubah data dari variable listKota, yang berisi daftar kota yang tersimpan di dalam *database*.
 3. public List<Organisasi> ListOrganisasi
 - *Property* ini digunakan untuk mengakses / mengubah data dari variable listOrganisasi, yang memuat organisasi yang tersimpan di dalam *database*.
 4. public List<Teman> ListTeman

- *Property* ini digunakan untuk mengakses / mengubah data dari variable listTeman, yang memuat daftar teman dari *current_user*.

```
#region ONLOAD
2 references
public void OnLoad()
{
    this.ListTeman = DAO_Teman.Select_ListTeman(this.Current_user.Username);
}
#endregion

#region METHOD (USER)
//For User
2 references
public void LogIn(string username, string password)
{
    this.Current_user = DAO_Users.User_Log_In(username, password);
}

2 references
public void Daftar(string username, string password, DateTime tglLahir, string noKTP, string foto, Kota kota)
{
    string newFotoPath = New_ProfilePictureFileName(username);
    File.Copy(foto, Path.Combine(this.MediafilePath, newFotoPath));
    User new_user = new User(username, password, tglLahir, noKTP, Path.Combine(this.MediafilePath,newFotoPath), kota);
    DAO_Users.User_Daftar(username, password, tglLahir, noKTP, Path.Combine(this.MediafilePathDB, newFotoPath), kota);
    this.Current_user = new_user;
}

#endregion
```

- Bagian *Method* :
 - public void OnLoad()
 - Digunakan untuk memberi nilai berupa daftar teman dari *current_user* pada variable listTeman melalui *property* ListTeman.
 - *Method* untuk Data *User*
 1. public void LogIn
 - Digunakan untuk memberi nilai pada *variable current_user* melalui *property Current_user*. Nilai didapatkan dengan meneruskan data *username* dan *password* ke *class DAO_Users* untuk diproses lebih lanjut, yaitu proses pengecekan di *database*.
 2. public void Daftar
 - Pertama, *method* ini digunakan untuk membuat *path* dari data foto yang dikirimkan. Selanjutnya, akan dilakukan proses penyimpanan data dengan bantuan *class entity User*. Data ini akan diberikan pada *variable current_user*

melalui *property Current_user*. Terakhir, data-data tersebut juga akan dikirimkan ke *class DAO_Users* untuk diteruskan ke dalam *database*.

```
#region METHOD (KISAH HIDUP)
//For Kisah Hidup
1 reference
public void Tambah_KisahHidup(Organisasi organisasi, int thawal, int thakhir, string deskripsi)
{
    KisahHidup newKisahHidup = new KisahHidup(organisasi, thawal, thakhir, deskripsi);
    DAO_KisahHidup.Insert_KisahHidup(newKisahHidup, this.Current_user);
    this.current_user.ListKisahHidup = DAO_KisahHidup.Select_ListKisahHidup(this.Current_user.Username);
}

endregion

#region METHOD (ORGANISASI)
1 reference
public void Tambah_Organisasi(Organisasi newOrganisasi)
{
    DAO_Organisasi.Insert_Organisasi(newOrganisasi);
    this.ListOrganisasi = DAO_Organisasi.Select_Organisasi(); //Re-New the list
}

1 reference
public List<Organisasi> Lihat_Organisasi_User()
{
    List<Organisasi> result = new List<Organisasi>();
    foreach (KisahHidup kh in this.Current_user.ListKisahHidup)
    {
        result.Add(kh.Organisasi);
    }
    return result;
}
#endregion
```

- *Method* untuk Data Kisah Hidup
 1. public void Tambah_KisahHidup
 - Digunakan untuk mengkapsulasi data-data yang ada melalui *class Entity KisahHidup*. *Entity* ini akan dikirimkan ke *class DAO_KisahHidup* untuk menjalani proses *insert* ke *database*. Tidak hanya itu, *method* ini juga digunakan untuk memberi nilai pada *Property ListKisahHidup* dari *variable current_user*. Nilai yang akan diberikan, didapatkan dari mengakses *method select* di dalam *class DAO_KisahHidup*.
- *Method* untuk Data Organisasi
 1. public void Tambah_Organisasi
 - Digunakan untuk menambahkan data organisasi baru ke *database* dengan bantuan *method* yang ada di dalam *class DAO_Organisasi*. Tidak hanya itu, nilai

dari *variable* listOrganisasi juga akan diperbarui melalui *method* di dalam *class* DAO_Organisasi.

2. public List<Organisasi> Lihat_Organisasi_User

- Digunakan untuk mengembalikan hasil berupa sebuah *list* organisasi yang pernah diikuti oleh *user*. Data organisasi yang pernah diikuti didapatkan dengan mengakses *property* ListKisahHidup dari *current_user* terlebih dahulu. Setelah itu, data organisasi didapatkan dengan mengakses *property* Organisasi dari masing-masing *object* KisahHidup di dalam *list*.

```
#region METHOD (TEMAN)
1 reference
public List<User> Cari_Teman(Organisasi organisasi)
{
    return DAO_Users.Select_ListUser_ByOrganisasi(organisasi, this.Current_user);
}

1 reference
public List<User> Cari_Teman(string username)
{
    return DAO_Users.Select_ListUser_ByUSN(username);
}

1 reference
public User Cari_AkunTeman(string username)
{
    return DAO_Users.Select_User(username);
}

1 reference
public void Request_Pertemanan(string username)
{
    DAO_Teman.Insert_RequestPertemanan(this.Current_user.Username, username);
}

1 reference
public void Terima_Pertemanan(string username)
{
    DAO_Teman.Update_RequestPertemanan(username, this.Current_user.Username, "Berteman");
    this.ListTeman = DAO_Teman.Select_ListTeman(this.Current_user.Username);
}

1 reference
public void Tolak_Pertemanan(string username)
{
    DAO_Teman.Update_RequestPertemanan(username, this.Current_user.Username, "Ditolak");
}

1 reference
public void KirimUlang_Pertemanan(string username)
{
    DAO_Teman.Update_RequestPertemanan(username, this.Current_user.Username, "Menunggu");
}

2 references
public List<Teman> Request_Pertemanan(bool jenis)
{
    //true = sender, false = receiver
    return DAO_Teman.Select_RequestPertemanan(this.Current_user.Username, jenis);
}
#endregion
```

- *Method* untuk Teman / Pertemanan

1. public List<User> Cari_Teman(Organisasi organisasi)

- Digunakan untuk mengembalikan hasil berupa sebuah *list user* yang berada di dalam satu organisasi bersama *current_user*. *List user* didapatkan dengan

memanfaatkan *method* yang ada di *class DAO_Users*, dengan mengirimkan parameter berupa organisasi dan *user* yang bersangkutan.

2. `public List<User> Cari_Teman(string username)`
 - Digunakan untuk mencari *user* lain berdasarkan *username* yang telah diinputkan, dikembalikan dalam bentuk `List<User>`. Proses pencarian *user* dilakukan dengan memanggil *method* yang ada di *class DAO_Users*, dengan mengirimkan parameter berupa *username* yang ingin dicari.
3. `public User Cari_AkunTeman(string username)`
 - Digunakan untuk mencari akun dari *user* lain berdasarkan *username* yang telah diinputkan, dikembalikan dalam bentuk *User*. Proses pencarian akun *user* dilakukan dengan memanggil *method* yang ada di *class DAO_Users*, dengan mengirimkan parameter berupa *username* yang ingin dicari.
4. `public void Request_Pertemanan(string username)`
 - Digunakan untuk memasukkan data *request* pertemanan ke dalam *database* melalui *method* yang ada di *class DAO_Teman*. Data yang dikirimkan ke *class DAO_Teman* adalah *username* dari *current_user* dan *username* ingin dijadikan teman.
5. `public void Terima_Pertemanan(string username)`
 - Digunakan untuk mengubah status pertemanan menjadi “Berteman” ketika *username* yang sebelumnya dituju telah menyetujui *request* yang dikirimkan. Proses *update* dilakukan dengan memanggil *method* yang ada di *class DAO_Teman*. Jika dicermati, peletakan *username* dari *current_user* dengan *username* yang tadinya dituju, menjadi terbalik, hal ini dikarenakan perubahan pada status pertemanan hanya bisa dilakukan jika *user* yang dituju telah menyetujui *request*. Terakhir, *variable* *listTeman* diperbarui / *di-refresh* melalui *method* untuk mencari list teman di *class DAO_Teman*.
6. `public void Tolak_Pertemanan(string username)`
 - *Method* ini memiliki konsep yang sama dengan sebelumnya, perbedaannya yaitu *user* yang dituju memilih untuk menolak *request* yang dikirimkan oleh *current_user*. Karena penolakan ini, *current_user* dengan *user* yang dituju tidak menjadi teman, sehingga tidak ada perubahan / *update* pada list teman.

7. public void KirimUlang_Pertemanan(string username)

- *Method* ini memiliki konsep yang sama dengan *method* Terima_Pertemanan, perbedaannya yaitu *user* yang dituju tidak membuat keputusan antara menerima atau menolak. Karena tidak adanya jawaban, hanya akan dilakukan perubahan pada status pertemanan, dimana statusnya menjadi “Menunggu”.

8. public void Request_Pertemanan(bool jenis)

- Digunakan untuk mencari data *request* pertemanan terkait *current_user*. Apabila *bool* jenis yang berada di parameter bernilai *true*, maka akan dilakukan pencarian *request* pertemanan yang dikirim oleh *current_user*. Sebaliknya, apabila bernilai *false*, akan dilakukan pencarian *request* pertemanan yang diterima oleh *current_user*. Pencarian *request* pertemanan dilakukan melalui *method* yang ada di *class DAO_Users*.

```
#region METHOD (KONTEN)
1 reference
public Konten Tambah_Komen(Komen komen, Konten newKonten)
{
    DAO_Komen.Insert_Komen(komen, newKonten.Id);
    newKonten.Comment = DAO_Komen.Select_Komen(newKonten.Id);
    return newKonten;
}

1 reference
public void Tambah_Tag(string username)
{
    this.buffer_konten.Tag.Add(DAO_Users.Select_User(username));
}

1 reference
public Konten Lihat_Konten(int id)
{
    return DAO_Konten.Select_Konten(id);
}

1 reference
public void Initiate_Konten()
{
    this.buffer_konten = new Konten();
}
```

- *Method* untuk Konten

1. public Konten Tambah_Komen(Komen komen, Konten newKonten)

- Digunakan untuk mengembalikan hasil berupa konten yang telah *diupdate* pada bagian *comment-nya*. Proses dimulai dengan memasukkan komen yang ditujukan pada satu konten khusus ke *database*. Setelah komen dimasukkan ke *database*, akan dilakukan pengambilan data komen berdasarkan *ID* dari konten.

Data-data komen ini akan diberikan kepada *property Comment* dari konten tersebut. Proses *input* dan pengambilan data dilakukan melalui *method* di *class DAO_Komen*.

2. `public void Tambah_Tag(string username)`

- Digunakan untuk menambahkan data *tag* pada konten. Proses penambahan dilakukan dengan mengambil data *user* berdasarkan *username* yang dicantumkan, terlebih dahulu melalui *method* di *DAO_Users*. Setelah itu, data *user* yang bersangkutan akan ditambahkan ke *property Tag* (berupa *List<Komen>*) dari konten yang berada di dalam *variable buffer_konten*.

3. `public Konten Lihat_Konten(int id)`

- Digunakan untuk mengembalikan data Konten berdasarkan *ID* yang diinputkan. Pencarian konten dilakukan dengan bantuan *method* dari *class DAO_Konten*.

4. `public Konten Initiate_Konten()`

- Digunakan untuk menambahkan nilai pada *variable buffer_konten*, untuk menghindari terjadinya *error unassigned local variable*.

```
1 reference
public void Tambah_Konten(string caption, OpenFileDialog fdialog)
{
    buffer_konten.Caption = caption;
    buffer_konten.TglUpload = DateTime.Now;
    string newPath = "";
    if (Path.GetExtension(fdialog.FileName) == ".jpg")
    {
        newPath = New_FileName(true);
        buffer_konten.Foto = Path.Combine(this.MediafilePathDB, newPath);
    }
    else
    {
        newPath = New_FileName(false);
        buffer_konten.Video = Path.Combine(this.MediafilePathDB, newPath);
    }
    File.Copy(fdialog.FileName, Path.Combine(this.MediafilePath, newPath));
    DAO_Konten.Insert_Konten(buffer_konten, this.Current_user.Username);
    this.Current_user.ListKonten = DAO_Konten.Select_ListKonten(this.current_user.Username);
    int konten_id = this.Current_user.ListKonten[(this.Current_user.ListKonten.Count - 1)].Id;
    foreach(User us in this.buffer_konten.Tag)
    {
        DAO_Tag.Insert_Tag(konten_id, us.Username);
    }
    this.buffer_konten = null;
}
```

5. public void Tambah_Konten(string caption, OpenFileDialog fdialog)

- Digunakan untuk menambahkan konten, lengkap dengan pembuatan komponen-komponennya. Proses dimulai dengan menambahkan *caption* dan tanggal upload pada konten yang bersangkutan. Selanjutnya, akan dibuat *path* untuk foto dan video yang ditambahkan. Setelah itu, akan dilakukan penambahan konten ke *database* dengan *current_user* sebagai pemiliknya. Setelah ditambahkan, listKonten dari *current_user* akan diperbarui kembali. Proses penambahan & *update* ini dilakukan dengan bantuan *method* dari *class DAO_Konten*.
- Setelah itu, perlu dilakukan penambahan data *tag* yang ada ke *database*. Untuk itu, diperlukan *ID* dari konten yang bersangkutan dengan mengakses listKonten yang telah diperbarui sebelumnya. Setelah mendapatkan *ID*, data para *user* yang di-*tag* akan ditambahkan ke *database* dengan *ID* konten selaku *identifier* melalui *method* di *class DAO_Tag*. Terakhir, variable *buffer_konten* dibuat *null* agar bisa menampung konten baru / yang akan diubah nantinya.

```
#region METHOD
//File Handling
2 references
private string New_FileName(bool type)
{
    string path = this.current_user.Username + "x" + DateTime.Now.ToString("yyyyMMddHHmmss") + "x" + DAO_Konten.Get_NewKonten_Id();
    if (type)
    {
        path += ".jpg";
    }
    else
    {
        path += ".mp4";
    }
    return path;
}

1 reference
private string New_ProfilePictureFileName(string username)
{
    return username + "xPFPx" + DateTime.Now.ToString("yyyyMMddHHmmss")+".jpg";
}

1 reference
private void CreateDirectory()
{
    if (!Directory.Exists(MediafilePath))
    {
        Directory.CreateDirectory(MediafilePath);
    }
}
#endregion
```

- *Method* untuk *File Handling*

1. private string New_FileName(bool type)

- Digunakan untuk mengembalikan hasil berupa *filename* untuk foto / video yang ditambahkan ke dalam konten. *Filename* yang dibuat terdiri dari *username*, tanggal penambahan, dan *ID* dari konten agar mudah untuk dilacak / dipahami

nantinya. Parameter *bool type* digunakan untuk menentukan jenis *extension* yang akan ditambahkan. Apabila bernilai *true*, maka akan ditambahkan *extension* .jpg untuk foto. Sebaliknya, akan ditambahkan *extension* .mp4 untuk video.

2. private string New_ProfilePictureFileName(string username)
 - *Method* ini memiliki konsep yang sama dengan *method* sebelumnya, perbedaannya ialah *filename* yang ini terdiri dari *username* dan tanggal penambahan. *Extension* yang ditambahkan sudah pasti .jpg karena berbentuk foto.
3. private void CreateDirectory()
 - *Method* ini memiliki konsep yang sama dengan *method* sebelumnya, perbedaannya ialah *filename* yang ini terdiri dari *username* dan tanggal penambahan. *Extension* yang ditambahkan sudah pasti .jpg karena berbentuk foto.

2.2 Class KoneksiDatabase

```
43 references
public class KoneksiDatabase
{
  #region DATA MEMBER
  private MySqlConnection koneksi;
  #endregion

  #region CONSTRUCTOR
  2 references
  public KoneksiDatabase(string server, string database, string uid, string pwd)
  {
    string ConnString = "Server=" + server + ";Database=" + database + ";Uid=" + uid + ";Pwd=" + pwd + ";";
    Koneksi = new MySqlConnection();
    this.Koneksi.ConnectionString = ConnString;
    Hubungkan_Database();
  }
  2 references
  public KoneksiDatabase()
  {
    string ConnString = "Server= localhost;Database=pameryuk;Uid=root;Pwd=";
    Koneksi = new MySqlConnection();
    this.Koneksi.ConnectionString = ConnString;
    Hubungkan_Database();
  }
  #endregion

  #region PROPERTIES
  //Properties yang akan diakses oleh luar
  9 references
  public MySqlConnection Koneksi { get => koneksi; private set => koneksi = value; }
  #endregion
```

- Bagian *Variables* :
 1. private MySqlConnection koneksi;

- *Variable* ini digunakan untuk menampung koneksi yang akan dibuat. *Variable* ini dibuat *private* karena nilainya akan diakses melalui *Property*
- Bagian *Constructor*:
 1. public KoneksiDatabase()
 - *Constructor* ini dipanggil untuk membuat koneksi dengan *server*, *database*, *Uid*, dan *password* yang sudah *fix* / pasti. Apabila ingin berpindah ke yang lain, maka yang harus diubah ialah *variable ConnString*.
 2. public KoneksiDatabase(string server, string database, string uid, string pwd)
 - Berbeda dengan yang sebelumnya, perubahan nilai pada *constructor* ini dapat dilakukan dengan mudah, Kita hanya perlu untuk mengganti nilai-nilai yang ada di parameter saja.
- Bagian *Property* :

1. public MySqlConnection Koneksi

- *Property* ini digunakan untuk mengakses data dari *variable* koneksi.

```
#region METHOD
/// <summary>
/// 
/// </summary>
/// <returns></returns>
1 reference
private bool Status_Database()
{
    //Mengecek koneksi dengan database terbuka atau tertutup, jika terbuka ditutup terlebih dahulu.
    return Koneksi.State == System.Data.ConnectionState.Open;
}

/// <summary>
/// 
/// </summary>
2 references
private void Hubungkan_Database()
{
    //Jika koneksi dengan database terbuka
    if(Status_Database())
    {
        //Tutup koneksi
        Koneksi.Close();
    }
    //Buka koneksi kembali
    Koneksi.Open();
}
```

```

21 references
public static MySqlDataReader DatabaseQueryCommand(string command)
{
    MySqlCommand cmd = new MySqlCommand();
    KoneksiDatabase c = new KoneksiDatabase();
    cmd.Connection = c.Koneksi;
    cmd.CommandText = command;
    MySqlDataReader dataReader = cmd.ExecuteReader();
    return dataReader;
}

/// <summary>
///
/// </summary>
/// <param name="command"></param>
12 references
public static void DatabaseDMLCommand(string command)
{
    MySqlCommand cmd = new MySqlCommand();
    KoneksiDatabase c = new KoneksiDatabase();
    cmd.Connection = c.Koneksi;
    cmd.CommandText = command;
    cmd.ExecuteNonQuery();
}

#endregion

```

- Bagian *Method* :
 1. private bool Status_Database()
 - Digunakan untuk mengecek status dari koneksi saat ini.
 2. private bool Hubungkan_Database()
 - Proses dimulai dengan memeriksa status koneksi terlebih dahulu. Apabila koneksi masih terbuka, maka koneksi harus ditutup terlebih dahulu. Di akhir, koneksi akan dibuka agar dapat terhubung ke *database*.
 3. public static MySqlDataReader DatabaseQueryCommand(string command)
 - *Method* inilah yang nantinya akan digunakan untuk proses pengambilan data di *class-class* yang lain. Karena *method* ini akan sering dipanggil, maka kami membuat *method* ini menjadi *static* agar tidak perlu berulang kali membuat *object* Koneksi demi mengakses *method* ini. Data-data akan dikembalikan dalam bentuk MySqlDataReader.
 4. public static void DatabaseQueryCommand(string command)
 - *Method* ini juga akan digunakan berulang kali untuk proses perubahan data di *database*. *Method* ini akan digunakan untuk menjalankan *DML Commands* yang terdiri dari INSERT, UPDATE, dan DELETE. Karena *method* ini akan sering dipanggil, maka kami membuat *method* ini menjadi *static* agar tidak perlu

berulang kali membuat *object* Koneksi demi mengakses *method* ini. Perubahan hanya akan terjadi di *database*, sehingga tidak ada data yang perlu untuk dikembalikan.

2.3 Class Repositories

```
1 reference
public class Repositories
{
    private List<Teman> listTeman;

    0 references
    public Repositories()
    {
    }

    0 references
    public List<Teman> ListTeman
    {
        get => listTeman;
        set => listTeman = value;
    }
}
```

- Bagian *Variables* :
 1. private List<Teman> listTeman;
 - *Variable* ini digunakan untuk menampung daftar teman dari *current_user*. *Variable* ini dibuat *private* karena nilainya akan diakses melalui *Property*.
- Bagian *Constructor* :
- Bagian *Property* :
 1. public List<Teman> ListTeman;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* listTeman, yang memuat daftar teman dari *current_user*.

2.4 Class StatusPertemanan

```
namespace PamerYukLibrary.Value
{
    0 references
    public enum StatusPertemanan
    {
        Menunggu,
        Berteman,
        Ditolak,
        Diblokir
    }
}
```

- Pada *class* ini, akan disediakan 4 jenis status pertemanan yang nantinya akan digunakan di *database*. Jenis-jenisnya antara lain “Menunggu”, “Berteman”, “Ditolak”, dan “Diblokir”.

2.5 Class User

```
45 references
public class User
{
    #region MEMBERS
    private string username;
    private string password;
    private string namaLengkap;
    private DateTime tglLahir;
    private string noKTP;
    private string fotoDiri;
    private string fotoProfil;
    private string email;
    private Kota kota;
    private List<KisahHidup> listKisahHidup;
    private List<Konten> listKonten;
```

- Bagian *Variables* :
 - Berikut merupakan *variables-variables* yang digunakan dalam *class Entity User*. *Variable-variable* ini akan digunakan untuk menampung data-data diri dari *user*. *Variable-variable* ini dibuat *private* karena nilainya akan diakses melalui *Property*

```
#region CONSTRUCTORS
2 references
public User(string username, string password, string namaLengkap, DateTime tglLahir, string noKTP, string fotoDiri, string fotoProfil, string email, Kota kota)
{
    this.Username = username;
    this.Password = password;
    this.NamaLengkap = namaLengkap;
    this.TglLahir = tglLahir;
    this.NoKTP = noKTP;
    this.FotoDiri = fotoDiri;
    this.FotoProfil = fotoProfil;
    this.Email = email;
    this.Kota = kota;
    this.ListKonten = new List<Konten>();
    this.ListKisahHidup = new List<KisahHidup>();
}

5 references
public User(string namaLengkap, DateTime tglLahir, string noKTP, string fotoDiri, string fotoProfil, string email, Kota kota)
{
    this.Username = null;
    this.Password = null;
    this.NamaLengkap = namaLengkap;
    this.TglLahir = tglLahir;
    this.NoKTP = noKTP;
    this.FotoDiri = fotoDiri;
    this.FotoProfil = fotoProfil;
    this.Email = email;
    this.Kota = kota;
    this.ListKonten = new List<Konten>();
    this.ListKisahHidup = new List<KisahHidup>();
}
#endregion
```

- Bagian *Constructor*:

1. public User(string username, string password, string namaLengkap, DateTime tglLahir, string noKTP, string fotoDiri, string fotoProfil, string email, Kota kota)
 - *Constructor* ini dipanggil ketika pengguna melakukan *login* ke sistem. Untuk keperluan *login* selanjutnya, data *password* haruslah disimpan. Pengisian *variable-variable* yang telah kami bahas sebelumnya, akan dilakukan melalui *property*.

2. public User(string username, string namaLengkap, DateTime tglLahir, string noKTP, string fotoDiri, string fotoProfil, string email, Kota kota)
 - Berbeda dengan yang sebelumnya, *constructor* ini akan dipanggil untuk memuat data-data *user* selain *current user*. Karena *password* hanya boleh diketahui oleh pemilik, maka data *password* tidak akan kami diakses maupun disimpan.

```
#region PROPERTIES
32 references
public string Username { get => username; set => username = value; }
2 references
public string Password { get => password; set => password = value; }
4 references
public string NamaLengkap { get => namaLengkap; set => namaLengkap = value; }
6 references
public DateTime TglLahir { get => tglLahir; set => tglLahir = value; }
5 references
public string NoKTP { get => noKTP; set => noKTP = value; }
5 references
public string FotoDiri { get => fotoDiri; set => fotoDiri = value; }
5 references
public string FotoProfil { get => fotoProfil; set => fotoProfil = value; }
4 references
public string Email { get => email; set => email = value; }
6 references
public Kota Kota { get => kota; set => kota = value; }
12 references
public List<KisahHidup> ListKisahHidup { get => listKisahHidup; set => listKisahHidup = value; }
11 references
public List<Konten> ListKonten { get => listKonten; set => listKonten = value; }
#endregion
```

- Bagian *Property* :
 1. public string Username;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* *username*, yang memuat nama akun yang digunakan oleh *user*.

 2. public string Password;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* *password*, yang memuat data semacam kunci yang telah dibuat oleh *user* untuk proses *login*.

3. public string NamaLengkap;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* namaLengkap, yang memuat nama lengkap dari *user*.
4. public DateTime TglLahir;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* tglLahir, yang memuat data tanggal lahir dari *user*.
5. public string NoKTP;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* noKTP, yang memuat data nomor KTP dari *user*.
6. public string FotoDiri;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* fotoDiri, yang memuat foto dari diri *user*.
7. public string FotoProfil;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* fotoProfil, yang memuat foto profil dari akun *user*.
8. public string Email;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* email, yang memuat data email yang digunakan oleh *user*.
9. public Kota Kota;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* kota, yang memuat data kota yang ditinggali saat ini / kota domisili dari *user*.
10. public List<KisahHidup> ListKisahHidup;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* listKisahHidup, yang memuat daftar kisah hidup yang dimiliki oleh *user*.
11. public List<Konten> ListKonten;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* listKonten, yang memuat daftar konten yang telah dibuat oleh *user*.

2.6 Class Kota

```
public class Kota
{
    private int id;
    private string nama;
```

- Bagian *Variables*:

1. private int id;

- *Variable* ini digunakan untuk menyimpan *index* untuk tiap kota yang ada.

2. private string nama;

- *Variable* ini digunakan untuk menyimpan data berupa nama dari kota yang bersangkutan.

```
9 references
public Kota(int id, string nama)
{
    this.Id = id;
    this.Nama = nama;
}
```

- Bagian *Constructor*:

- *Property* this.Id

- this.Id dipanggil ketika melakukan proses pendaftaran *user* baru untuk memberikan sebuah *index* yang berbeda-beda untuk tiap kota yang ada.

- *Property* this.Nama

this.Nama dipanggil ketika melakukan pendaftaran *user* baru, this.Nama menggunakan tipe string untuk menyimpan nama kota.

```
public int Id
{ get => id;
  set
  {
    if(value<0)
    {
      throw new Exception("ID can't be lower than 0");
    }
    else
    {
      id = value;
    }
  }
}
5 references
public string Nama
{
  get => nama;
  set
  {
    if(value=="")
    {
      throw new Exception("Kota name can't be empty");
    }
    else
    {
      nama = value;
    }
  }
}
```

```
0 references
public override string ToString()
{
    return this.Nama;
}
```

- Bagian *Property*:

1. public int Id:

- Property ini digunakan untuk mengakses / mengubah data dari variable id yang membuat daftar kode unik pada setiap kota. Selain itu, kode unik diharuskan memiliki nilai lebih dari 0.

2. public string Nama:

- Property ini digunakan untuk mengakses / mengubah data dari variable nama yang memuat daftar semua kota. Data yang akan dimasukkan ke *variable* nama hendaknya tidak boleh kosong.

- Bagian *Method*:

1. public override string ToString()

- *Method* ini akan dipanggil ketika terdapat *object* Kota yang akan diubah menjadi *string*. Dengan adanya *method* ini, keambiguan dapat dicegah dan data yang akan dikembalikan adalah nama dari *object* Kota yang bersangkutan.

2.7 Class Organisasi

```
30 references
public class Organisasi
{
    private int id;
    private string nama;
    private Kota kota;

    2 references
    public Organisasi(int id, string nama, Kota kota) //From DB
    {
        this.Id = id;
        this.Nama = nama;
        this.Kota = kota;
    }

    1 reference
    public Organisasi(string nama, Kota kota) //New Insert
    {
        this.Nama = nama;
        this.Kota = kota;
    }

    1 reference
    public Organisasi()
    {
        this.Id = -1;
        this.Nama = "";
        this.Kota = null;
    }
}
```

- Bagian *Variables* :

1. private int id;

- *Variable* ini digunakan untuk menyimpan *index* untuk tiap organisasi yang ada.

2. private string nama;

- *Variable* ini digunakan untuk menyimpan data berupa nama dari organisasi yang bersangkutan.

- 3. private Kota kota;
 - *Variable* ini digunakan untuk menyimpan data berupa kota dari organisasi yang bersangkutan.
- Bagian *Constructor*:
 1. public Organisasi(int id, string nama, Kota kota)
 - *Constructor* ini dipanggil ketika terjadi proses *input* data organisasi ke *database* untuk yang pertama kalinya / proses pengambilan data dari *database*.
 2. public Organisasi(string nama, Kota kota)
 - Berbeda dengan yang sebelumnya, *constructor* ini akan dipanggil ketika terjadi proses penambahan data ke tabel yang sudah terisi. *Variable id* tidak perlu disertakan karena tersedia fitur *auto increment* yang akan secara otomatis mengisi bagian *id*.
 3. public Organisasi()
 - Berbeda dengan yang lainnya, *constructor* ini akan hanya akan dipanggil sewaktu inisialisasi awal saja.

```
4 references
public int Id { get => id; set => id = value; }
5 references
public string Nama { get => nama; set => nama = value; }
4 references
public Kota Kota { get => kota; set => kota = value; }
```

- Bagian *Property* :
 1. public int Id;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable id*, yang memuat *index* dari suatu organisasi.
 2. public string Nama;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable nama*, yang memuat data berupa nama dari suatu organisasi.

3. public Kota Kota;

- *Property* ini digunakan untuk mengakses / mengubah data dari *variable* kota, yang memuat kota dari suatu organisasi.

2.8 Class Teman

```
20 references
public class Teman
{
    private string username;
    private DateTime tglBerteman;
    private string status;
}
4 references
```

- Bagian *Variables*:
 1. private string username
 - *Variable* ini digunakan untuk menyimpan nama akun dari teman.
 2. private DateTime tglBerteman:
 - *Variable* ini digunakan untuk menyimpan tanggal saat pertemanaan di mulai.
 3. private string status:
 - *Variable* ini digunakan untuk menyimpan status pertemanaan, contohnya: Berteman, Diblokir, dan lainnya.

```
4 references
public Teman(string username, DateTime tglBerteman, string status)
{
    this.Username = username;
    this.TglBerteman = tglBerteman;
    this.Status = status;
}
```

• Bagian *Constructor*:

1. public Teman (string username, DateTime tglBerteman, string status)
 - *Constructor* ini dipanggil ketika melakukan proses penambahan teman. Tipe data string digunakan untuk menyimpan username dan status pertemanaan, sedangkan tipe data DateTime digunakan untuk menyimpan tanggal pertemanaan dimulai.

```
3 references
public string Username { get => username; set => username = value; }
1 reference
public DateTime TglBerteman { get => tglBerteman; set => tglBerteman = value; }
1 reference
public string Status { get => status; set => status = value; }
```

- Bagian *Property*:
 1. public string Username
 - Property ini digunakan untuk mengatur dan mengambil nilai dari variable username yang memuat data nama teman.
 2. public DateTime TglBerteman
 - Property ini digunakan untuk mengatur dan mengambil nilai dari variable TglBerteman.
 3. public Status
 - Property ini digunakan untuk mengatur dan mengambil nilai dari variable Status.

2.9 Class KisahHidup

```

15 references
public class KisahHidup
{
    private Organisasi organisasi;
    private int thawal;
    private int thakhir;
    private string deskripsi;

    2 references
    public KisahHidup(Organisasi organisasi, int thawal, int thakhir, string deskripsi)
    {
        this.Organisasi = organisasi;
        this.Thawal = thawal;
        this.Thakhir = thakhir;
        this.Deskripsi = deskripsi;
    }

    4 references
    public Organisasi Organisasi { get => organisasi; set => organisasi = value; }
    2 references
    public int Thawal { get => thawal; set => thawal = value; }
    2 references
    public int Thakhir { get => thakhir; set => thakhir = value; }
    2 references
    public string Deskripsi { get => deskripsi; set => deskripsi = value; }

    1 reference
    public override string ToString()
    {
        string text = "";
        text = this.Organisasi.Nama + " dari tahun " + this.thawal + " hingga " + this.thakhir + "\n";
        return text;
    }
}

```

- Bagian *Variables* :
 1. private Organisasi organisasi;
 - *Variable* ini digunakan untuk menyimpan data organisasi yang pernah diikuti oleh *user*.

2. private int thawal;
 - *Variable* ini digunakan untuk menyimpan data berupa tahun di mana *user* mulai memasuki organisasi tersebut.
 3. private int thakhir;
 - *Variable* ini digunakan untuk menyimpan data berupa tahun di mana *user* lulus / keluar dari organisasi yang bersangkutan.
 4. private string deskripsi;
 - *Variable* ini digunakan untuk menyimpan data berupa deskripsi pengalaman dari *user* sewaktu berada di dalam organisasi yang bersangkutan.
- Bagian *Constructor*:
 1. public KisahHidup(Organisasi organisasi, int thawal, int thakhir, string deskripsi)
 - *Constructor* ini dipanggil untuk memasukkan data-data yang ada ke *variable* yang bersesuaian melalui *Property*.
 - Bagian *Property* :
 1. public Organisasi Organisasi;
 - *Property* ini digunakan untuk mengakses / mengubah data dari organisasi, yang memuat data organisasi yang diikuti oleh *user*.
 2. public int Thawal;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* thawal, yang memuat data berupa tahun di mana *user* mulai memasuki organisasi tersebut.
 3. public int Thakhir;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* thakhir, yang memuat data berupa tahun di mana *user* lulus / keluar dari organisasi yang bersangkutan.
 4. public string Deskripsi;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable* deskripsi yang memuat data berupa deskripsi pengalaman dari *user* sewaktu berada di dalam organisasi yang bersangkutan.
 - Bagian *Method* :
 1. public override string ToString()

- *Method* ini akan dipanggil ketika terdapat *object Organisasi* yang akan diubah menjadi *string*. Dengan adanya *method* ini, keambiguan dapat dicegah dan data yang akan dikembalikan adalah nama dari *object Organisasi* yang bersangkutan, beserta dengan detail tahun awal dan tahun akhir.

2.10 Class Konten

```
27 references
public class Konten
{
    private int id;
    private string caption;
    private string foto;
    private string video;
    private DateTime tglUpload;
    private List<User> tag;
    private int like;
    private List<Komen> comment;
```

- Bagian *Variables*:
 1. private int id;
 - *Variable* ini digunakan untuk menyimpan kode unik pada setiap konten.
 2. private string caption;
 - *Variable* ini digunakan untuk menyimpan teks ataupun deskripsi yang berkaitan dengan konten.
 3. private string foto;
 - *Variable* ini digunakan untuk menyimpan file foto yang nanti nya akan menjadi konten.
 4. private string video;
 - *Variable* ini digunakan untuk menyimpan file video yang nanti nya akan menjadi konten.
 5. private DateTime tglUpload;
 - *Variable* ini digunakan untuk menyimpan tanggal konten ketika diunggah.
 6. private List<User> tag;
 - *Variable* ini digunakan untuk menyimpan daftar pengguna yang di tandai (dalam bentuk List<User>)
 7. private int like
 - *Variable* ini digunakan untuk menyimpan jumlah suka (like) pada konten.
 8. private List<Komen> comment:

- *Variable* ini digunakan untuk menyimpan daftar komentar (dalam bentuk `List<Komen>`)

```

2 references
public Konten(int id, string caption, string foto, string video, DateTime tglUpload)
{
    this.Id = id;
    this.Caption = caption;
    this.Foto = foto;
    this.Video = video;
    this.TglUpload = tglUpload;
    this.Comment = DAO_Komen.Select_Komen(this.Id);
    this.Tag = DAO_Tag.Select_Tag(this.Id);
    this.Like = DAO_Like.Select_LikeCount(this.Id);
}

0 references
public Konten( string caption, string foto, string video, DateTime tglUpload) //New Konten
{
}
1 reference
public Konten() //Buffer Konten
{
    this.Caption = "caption";
    this.Foto = "null";
    this.Video = "null";
    this.Tag = new List<User>();
}

```

- Bagian *Constructor*
 1. `public Konten(int id, string caption, string foto, string video, DateTime tglUpload)`
 - *Constructor* ini digunakan untuk memberikan nilai ke *variable* konten, lengkap dengan *variable* yang lainnya juga. Sementara itu, untuk *variable* seperti komentar, tag, dan like, datanya akan diambil dari database melalui *class DAO*.
 2. `public Konten(string caption, string foto, string video, DateTime tglUpload)`
 - *Constructor* ini digunakan untuk membuat objek Konten baru yang belum memiliki ID, komentar, atau jumlah suka. Constructor ini dipanggil ketika membuat konten baru.
 3. `public Konten()`
 - Memberikan nilai *default*, seperti caption diatur ke "caption", foto dan video diatur ke "null", serta tag diinisialisasi sebagai daftar kosong.
- Bagian *Property*

```
12 references
public int Id { get => id; set => id = value; }
6 references
public string Caption { get => caption; set => caption = value; }
7 references
public string Foto { get => foto; set => foto = value; }
4 references
public string Video { get => video; set => video = value; }
4 references
public DateTime TglUpload { get => tglUpload; set => tglUpload = value; }
4 references
public List<User> Tag { get => tag; set => tag = value; }
2 references
public int Like { get=>like; set => like = value; }
3 references
public List<Komen> Comment { get => comment; set => comment = value; }
```

1. public int Id:
 - *Property* ini digunakan untuk mengatur dan mengambil nilai Id pada konten.
2. public string Caption:
 - *Property* ini digunakan untuk mengakses teks atau deskripsi pada konten.
3. public string Foto:
 - *Property* ini digunakan untuk mengakses dan mengatur file foto.
4. public string Video:
 - *Property* ini digunakan untuk mengakses dan mengatur file video.
5. public DateTime TglUpload:
 - *Property* ini digunakan untuk mengakses dan mengatur tanggal unggah.
6. public List<User> Tag:
 - Property ini digunakan untuk mengakses dan mengatur daftar pengguna yang ditandai
7. public int Like
 - Property ini digunakan untuk mengatur dan mengakses jumlah suka pada konten.
8. public list<Komen> Comment:
 - Property ini digunakan untuk mengakses dan mengatur daftar komentar yang ada pada konten.

2.11 Class Komen

```
13 references
public class Komen
{
    private int id;
    private string komentar;
    private DateTime tgl;
    private string user;

    1 reference
    public Komen(int id, string komentar, DateTime tgl, string user)
    {
        this.Id = id;
        this.Komentar = komentar;
        this.Tgl = tgl;
        this.User = user;
    }

    1 reference
    public Komen(string komentar, DateTime time, string user) //New Comment
    {
        this.Komentar = komentar;
        this.Tgl = time;
        this.User = user;
    }

    1 reference
    public int Id { get => id; set => id = value; }
    4 references
    public string Komentar { get => komentar; set => komentar = value; }
    3 references
    public DateTime Tgl { get => tgl; set => tgl = value; }
    4 references
    public string User { get => user; set => user = value; }

    0 references
    public override string ToString()
    {
        return this.tgl + " -> " + this.User + " : " + this.Komentar;
    }
}
```

- Bagian *Variables* :
 1. private int id;
 - *Variable* ini digunakan untuk menyimpan data berupa *index* untuk tiap komen.
 2. private string komentar;
 - *Variable* ini digunakan untuk menyimpan data berupa tulisan komentar yang diketik oleh *user*.
 3. private DateTime tgl;
 - *Variable* ini digunakan untuk menyimpan data berupa tanggal komen itu dibuat.
 4. private string user;

- *Variable* ini digunakan untuk menyimpan data berupa siapa *user* yang membuat komentar tersebut.
- Bagian *Constructor*:
 1. public Komen(int id, string komentar, DateTime tgl, string user)
 - *Constructor* ini dipanggil ketika terjadi proses *input* data komen ke *database* untuk yang pertama kalinya / proses pengambilan data komen dari *database*.
 2. public KisahHidup(string komentar, DateTime tgl, string user)
 - Berbeda dengan yang sebelumnya, *constructor* ini akan dipanggil ketika terjadi proses penambahan data ke tabel komen yang sudah terisi. *Variable id* tidak perlu disertakan karena tersedia fitur *auto increment* yang akan secara otomatis mengisi bagian *id*.
- Bagian *Property* :
 1. public int Id;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable id*, yang memuat data berupa *index* dari komen tersebut.
 2. public string Komentar;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable komentar*, yang memuat data berupa tulisan komentar yang diketik oleh *user*.
 3. Public DateTime Tgl;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable tgl*, yang memuat data berupa berupa tanggal komen itu dibuat.
 4. public string User
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable user* yang memuat data berupa siapa *user* yang membuat komentar tersebut.
- Bagian *Method* :
 1. public override string ToString()
 - *Method* ini akan dipanggil ketika terdapat *object Komen* yang akan diubah menjadi *string*. Dengan adanya *method* ini, keambiguan dapat dicegah dan data

yang akan dikembalikan adalah tanggal pembuatan komen, *user* yang membuat, dan komentar yang diberikan dari *object* Komen yang bersangkutan.

Note : Class DAO hanya berisi Method saja. Variables, Constructor, dan Property sudah diatur di dalam class Entity yang telah kami jelaskan sebelumnya.

2.12 Class DAO_Users

Karena *method* di dalam *class DAO* akan sering diakses, maka kami membuat *method* yang ada di seluruh *class DAO* menjadi *static* agar tidak perlu repot membuat *object* demi mengakses *method* yang ada. Tidak hanya itu, penambahan sifat *static* akan mempermudah pemrosesan data ke *database*, dimana *method-method* ini dapat segera dipanggil setelah data-data yang ada di enkapsulasi di *class Entity*.

```
public static User User_Log_In(string username, string password)
{
    string perintah = "SELECT * FROM user u inner join kota k on u.kota_id = k.id where username ='" + username + "' and password ='" + password + "'";

    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(perintah);

    User user;
    if (dr.Read())
    {
        //User
        username = dr.GetValue(0).ToString();
        password = dr.GetValue(1).ToString();
        string namaLengkap = dr.GetValue(2).ToString();
        DateTime tglLahir = DateTime.Parse(dr.GetValue(3).ToString());
        string noKTP = dr.GetValue(4).ToString();
        string fotoDiri = dr.GetValue(5).ToString(); //Still confuse with this image data format
        string fotoProfil = dr.GetValue(6).ToString();
        string email = dr.GetValue(7).ToString();
        int kota_id_fk = int.Parse(dr.GetValue(8).ToString());

        //Kota
        int kota_id = int.Parse(dr.GetValue(9).ToString());
        string nama = dr.GetValue(10).ToString();

        //Create Kota
        Kota newKota = new Kota(kota_id, nama);

        //Create User
        user = new User(username, password, namaLengkap, tglLahir, noKTP, fotoDiri, fotoProfil, email, newKota);
        user.ListKonten = DAO_Konten.Select_ListKonten(username);
        user.ListKisahHidup = DAO_KisahHidup.Select_ListKisahHidup(username);
        Console.WriteLine("Data user dari database berhasil diambil");
    }
    else
    {
        throw new Exception("User password or username is incorrect");
    }
    return user;
}
```

- *Method* ini digunakan untuk mengambil data *user* berdasarkan *username* dan *password* yang diinputkan. Apabila data tidak ditemukan, maka *user* akan diberikan notifikasi *exception*. Data-data akan diambil dari tabel *user* yang digabungkan dengan tabel *kota*. Data-data terkait *user* akan disimpan di dalam *variable* untuk dibuat *object* nya melalui *constructor* yang ada di *class User*. Sementara data-data terkait *kota* (*id* & *nama kota*) akan dikirimkan ke *constructor* yang ada di *class Kota*. *ListKonten* dan *ListKisahHidup* dari *user* didapatkan dengan mengakses *method* di *class DAO_Konten* dan *DAO_KisahHidup*.

```
public static void User_Daftar(string username, string password, string namaLengkap,
DateTime tglLahir, string noKTP, string fotoDiri, string fotoProfil, string email, Kota kota)

{
    string command = "INSERT INTO `pameryuk`.`user` (`username`, `password`,
`namaLengkap`, `tglLahir`, `noKTP`, `fotoDiri`, `fotoProfil`, `email`, `Kota_id`)
VALUES('" + username + "', '" + password + "', '" + namaLengkap + "', '" + tglLahir.ToString("yyyy-MM-dd") + "', '" + noKTP + "', '" + fotoDiri + "', '" + fotoProfil + "', '" + email + "', '" + kota.Id + "');";

    KoneksiDatabase.DatabaseDMLCommand(command);
}
```

```
public static void Update_User(string current_username, string new_usn, string new_nama,
DateTime new_date, string new_ktp, string new_fd, string new_fp, string new_email, Kota
new_kota)

{
    try
    {
        string command = "UPDATE `pameryuk`.`user` SET `username`=''" + new_usn + "'",
`namaLengkap`=''" + new_nama + "'", `tglLahir`=''" + new_date.ToString("yyyy-MM-dd") + "'",
`noKTP`=''" + new_ktp + "'", `fotoDiri`=''" + new_fd + "'", `fotoProfil`=''" + new_fp + "'",
`email`=''" + new_email + "'", `Kota_id`=''" + new_kota.Id + "''
        WHERE `username`=''" + current_username + "'";;

        KoneksiDatabase.DatabaseDMLCommand(command);
    }
    catch (Exception ex)
    {
        throw new Exception("Update error : " + ex.Message);
    }
}
```

```

    }
}

}

```

- Kedua *method* ini digunakan untuk mengeksekusi perintah DML, yaitu INSERT dan UPDATE User. 9 dari 11 *variable* yang dimiliki oleh *class* User dapat ditambahkan / diubah melalui *method* ini, sesuai dengan yang bisa *diedit* di *interface* yang telah kami buat. Penambahan data terkait ListKonten dan ListKisahHidup akan diatur di *menu interface* yang lain, dengan bantuan dari *class* DAO_Konten dan DAO_KisahHidup.

```

1 reference
public static List<User> Select_ListUser_ByUSN(string username)
{
    string perintah = "SELECT * FROM user u inner join kota k on u.kota_id = k.id where username ='" + username + "'"; 

    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(perintah);
    List<User> listAkun = new List<User>();
    User user;
    while (dr.Read())
    {
        //User
        string usname = dr.GetValue(0).ToString();
        string namalengkap = dr.GetValue(2).ToString();
        DateTime tgllahir = DateTime.Parse(dr.GetValue(3).ToString());
        string noKTP = dr.GetValue(4).ToString();
        string fotoDiri = dr.GetValue(5).ToString(); //Still confuse with this image data format
        string fotoProfil = dr.GetValue(6).ToString();
        string email = dr.GetValue(7).ToString();
        int kota_id_fk = int.Parse(dr.GetValue(8).ToString());

        //Kota
        int kota_id = int.Parse(dr.GetValue(9).ToString());
        string nama = dr.GetValue(10).ToString();

        //Create Kota
        Kota newKota = new Kota(kota_id, nama);

        //Create User
        user = new User(usname, namalengkap, tgllahir, noKTP, fotoDiri, fotoProfil, email, newKota);
        user.ListKonten = DAO_Konten.Select_ListKonten(username);
        user.ListKisahHidup = DAO_KisahHidup.Select_ListKisahHidup(username);
        listAkun.Add(user);
        Console.WriteLine("Data user dari database berhasil diambil");
    }
    return listAkun;
}

```

- Konsep kerja dari *method* ini hampir sama dengan *method* User_Log_In. Yang menjadi pembeda yaitu *constructor* yang digunakan dan pembuatan *list* baru. Karena data yang dikembalikan merupakan data akun *user* lain, maka data *password* tidak akan diakses. Tidak hanya itu, dibuat sebuah *list* baru untuk menampung data-data *user* yang ada.

```

public static User Select_User(string username)
{
    string perintah = "SELECT * FROM user u inner join kota k on u.kota_id = k.id where username ='" + username + "'";

    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(perintah);
    User user;
    if (dr.Read())
    {
        //User
        string usname = dr.GetValue(0).ToString();
        string namalengkap = dr.GetValue(2).ToString();
        DateTime tgllahir = DateTime.Parse(dr.GetValue(3).ToString());
        string noKTP = dr.GetValue(4).ToString();
        string fotoDiri = dr.GetValue(5).ToString(); //still confuse with this image data format
        string fotoProfil = dr.GetValue(6).ToString();
        string email = dr.GetValue(7).ToString();
        int kota_id_fk = int.Parse(dr.GetValue(8).ToString());

        //Kota
        int kota_id = int.Parse(dr.GetValue(9).ToString());
        string nama = dr.GetValue(10).ToString();

        //Create Kota
        Kota newKota = new Kota(kota_id, nama);

        //Create User
        user = new User(usname, namalengkap, tgllahir, noKTP, fotoDiri, fotoProfil, email, newKota);
        user.ListKonten = DAO_Konten.Select_ListKonten(username);
        user.ListKisahHidup = DAO_KisahHidup.Select_ListKisahHidup(username);
        return user;
    }
    else return null;
}

```

- Konsep kerja dari *method* ini hampir sama dengan *method* sebelumnya. Yang membedakan ialah hasil yang dikembalikan dari *method* ini hanya 1 *user*.

```
public static List<User> Select_ListUser_ByOrganisasi(Organisasi organisasi, User current_user)
```

```
{
    string command = "SELECT u.username, u.namaLengkap, u.tgllahir, u.noktp, u.fotoDiri,
    u.fotoProfil, u.email, u.kota_id, ko.nama FROM user u inner join kisahhidup k on u.username
    = k.username inner join kota ko on u.kota_id = ko.id where k.organisasi_id = '"+organisasi.Id+
    "' and k.username != '"+current_user.Username+"';"
```

```
MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(command);
```

```
List<User> listAkun = new List<User>();
```

```
User user;
```

```
while (dr.Read())
```

```
{
```

```
//User

string usname = dr.GetValue(0).ToString();

string namaLengkap = dr.GetValue(2).ToString();

DateTime tglahir = DateTime.Parse(dr.GetValue(3).ToString());

string noKTP = dr.GetValue(4).ToString();

string fotoDiri = dr.GetValue(5).ToString(); //Still confuse with this image data format

string fotoProfil = dr.GetValue(6).ToString();

string email = dr.GetValue(7).ToString();

int kota_id_fk = int.Parse(dr.GetValue(8).ToString());


//Kota

string nama = dr.GetValue(9).ToString();


//Create Kota

Kota newKota = new Kota(kota_id_fk, nama);


//Create User

user = new User(usname,namaLengkap, tglahir, noKTP, fotoDiri, fotoProfil, email,
newKota);

user.ListKonten = DAO_Konten.Select_ListKonten(usname);

user.ListKisahHidup = DAO_KisahHidup.Select_ListKisahHidup(usname);

listAkun.Add(user);

Console.WriteLine("Data user dari database berhasil diambil");

}

return listAkun;
```

}

- *Method* ini digunakan untuk mengambil data dari *user-user* berdasarkan organisasi yang dipilih. Data-data akan diambil dari tabel *user* yang digabungkan dengan tabel kisah hidup lalu digabungkan dengan tabel kota. Data-data terkait *user* akan disimpan di dalam *variable* untuk dibuat *object* nya melalui *constructor* (tanpa data *password*) yang ada di *class* User. Sementara data-data terkait kota (id & nama kota) akan dikirimkan ke *constructor* yang ada di *class* Kota. ListKonten dan ListKisahHidup dari *user* didapatkan dengan mengakses *method* di *class* DAO_Konten dan DAO_KisahHidup. Data-data yang didapatkan akan dimasukkan ke dalam *list* dan akan dikembalikan sebagai hasil.

2.13 Class DAO_Kota

```
1 reference
public class DAO_Kota
{
    1 reference
    public static List<Kota> Select_ListKota()
    {
        string command = "SELECT * FROM kota;";
        MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(command);
        List<Kota> result=new List<Kota>();
        while(dr.Read())
        {
            int id = int.Parse(dr.GetValue(0).ToString());
            string nama = dr.GetValue(1).ToString();
            Kota newKota = new Kota(id,nama);
            result.Add(newKota);
        }
        return result;
    }
}
```

- Method ini digunakan untuk mengambil semua data dari *database*, khususnya tabel Kota. Setelah itu, *method* ini akan mengembalikan data, untuk ditampilkan ke dalam *interface*. *Method* ini mengembalikan daftar kota dalam bentuk *list*. Dengan ada nya method ini, pengambilan data secara otomatis dapat dilakukan tanpa menulis querry berulang kali.

2.14 Class DAO_Organisasi

```
3 references
public class DAO_Organisasi
{
    2 references
    public static List<Organisasi> Select_Organisasi()
    {

        string command;
        command = "SELECT * FROM organisasi inner join kota on organisasi.kota_id = kota.id";
        MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(command);
        List<Organisasi> result = new List<Organisasi>();
        while(dr.Read())
        {
            //Organisasi
            int id = int.Parse(dr.GetValue(0).ToString());
            string nama = dr.GetValue(1).ToString();
            int kota_id_fk = int.Parse(dr.GetValue(2).ToString());

            //Kota (join)
            int kota_id = int.Parse(dr.GetValue(3).ToString());
            string kota_nama = dr.GetValue(4).ToString();

            //Create Kota (buffer)
            Kota newKota = new Kota(kota_id,kota_nama);

            //Create Organisasi (buffer)
            Organisasi newOrganisasi = new Organisasi(id,nama,newKota);

            //Add to list
            result.Add(newOrganisasi);
        }
        return result;
    }
}
```

- *Method* ini digunakan untuk mengambil data organisasi dari *database*, yang nantinya akan ditampilkan di *interface*. Data-data akan diambil dari tabel organisasi yang dibungkus dengan tabel kota. Data-data terkait kota akan disimpan di dalam *variable* untuk dibuat *object* nya melalui *constructor* yang ada di *class* Kota. Sementara data-data terkait organisasi (id, nama, dan kota dari organisasi tersebut) akan dikirimkan ke *constructor* yang ada di *class* Organisasi. Data-data organisasi yang didapatkan akan dimasukkan ke dalam *list* dan akan dikembalikan sebagai hasil.

```
1 reference
public static void Insert_Organisasi(Organisasi organisasi)
{
    string command = "INSERT INTO organisasi ('id', 'nama', 'Kota_id') VALUES ('"+ new_id_organisasi().ToString() +"', '"+organisasi>Nama+"', '"+organisasi.Kota.Id+"')";
    KoneksiDatabase.DatabaseDMLCommand(command);
}

1 reference
private static int new_id_organisasi()
{
    string command = "SELECT o.id FROM organisasi o ORDER BY o.id DESC LIMIT 1";
    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(command);
    int result = -1;
    if (dr.Read())
    {
        result = int.Parse(dr.GetValue(0).ToString());
        result++;
    }
    return result;
}
```

- *Method* Insert_Organisasi digunakan untuk menambahkan organisasi baru ke *database*. Class organisasi sendiri tersusun dari 3 *variable* (id, nama, dan kota), dimana ke-3 data tersebut dapat ditambahkan melalui *method* ini.
- *Method* new_id_organisasi digunakan untuk menentukan *id* untuk organisasi yang akan ditambahkan. Cara kerja method ini adalah dengan mengakses *id* terbesar yang ada di dalam tabel organisasi. Dengan demikian, *id* untuk organisasi baru didapatkan dengan menambahkan nilai 1 pada *id* terbesar.

2.15 Class DAO_Teman

```
public static List<Teman> Select_ListTeman(string username)

{
    string perintah;

    perintah = "SELECT * FROM teman WHERE teman.username1 = '"+username+"' and
teman.StatusPertemanan = 'Berteman' UNION SELECT * FROM teman WHERE
teman.username2 = '"+username+" and teman.StatusPertemanan = 'Berteman';";

    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(perintah);

    List<Teman> listTeman = new List<Teman>();

    while (dr.Read())
    {
        //Teman

        string username1 = dr.GetValue(0).ToString();

        string username2 = dr.GetValue(1).ToString();

        DateTime tglBerteman = DateTime.Parse(dr.GetValue(2).ToString());

        string status = dr.GetValue(3).ToString();

        string friendUSN = username1;

        if(friendUSN == username)
        {

```

```

friendUSN = username2;

}

//Create Teman(buffer)

Teman newTeman = new Teman(friendUSN,tglBerteman,status);

//Add to list teman

listTeman.Add(newTeman);

}

return listTeman;
}

```

- *Method* ini digunakan untuk mengakses *list* teman dari *current_user* dari *database*, yang nantinya akan ditampilkan di *interface*. Data-data akan diambil dari tabel teman yang sudah memiliki status “Berteman”, baik itu dari *current_user* yang mengirimkan *request* maupun dari *current_user* yang menerima request. Untuk mencari data-data yang memenuhi kriteria tersebut, diperlukan *command UNION* untuk menggabungkan data-data yang ada. Data-data tersebut akan disimpan di dalam *variable* untuk dibuat *object* nya melalui *constructor* yang ada di *class Teman*. *Object-object* yang telah berisi data-data tersebut akan dimasukkan ke dalam *list* dan dikembalikan sebagai hasil.

```

public static void Insert_RequestPertemanan(string username1, string username2)

{
    string command = "INSERT INTO `pameryuk`.`teman` ('username1',
`username2`,`tglberteman`, `statusPertemanan`) VALUES ('"+username1+"',
"+username2+"','"++ DateTime.Now.Date.ToString("yyyy-MM-dd") +"','Menunggu');";

KoneksiDatabase.DatabaseDMLCommand(command);

}

```

```

public static void Update_RequestPertemanan(string username1, string username2, string
statusRequest)

{
    string command = "UPDATE teman set statusPertemanan = '"+statusRequest+"",
tglberteman = '"+DateTime.Now.Date.ToString("yyyy-MM-dd")+"' where username1 =
"+username1+" and username2='"+username2+"';";

    KoneksiDatabase.DatabaseDMLCommand(command);

}

```

- Kedua *method* ini digunakan untuk mengeksekusi perintah untuk menambahkan maupun mengubah status pertemanan. Untuk bagian penambahan, tanggal pertemanan akan di-set menjadi waktu saat *method* itu diakses dan status pertemanan akan otomatis di-set menjadi menunggu. Sementara untuk bagian update, tanggal pertemanan juga akan di-set menjadi waktu saat *method* itu diakses dan status pertemanan akan diubah sesuai dengan parameter yang dikirimkan.

```

1 reference
public static List<Teman> Select_RequestPertemanan(string username, bool jenis)
{
    string perintah;
    if(jenis == true)
        perintah = "SELECT * FROM teman WHERE teman.username1 = '" + username + "' and teman.StatusPertemanan != 'Berteman'";
    else
        perintah = "SELECT * FROM teman WHERE teman.username2 = '" + username + "' and teman.StatusPertemanan != 'Berteman'";

    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(perintah);
    List<Teman> listTeman = new List<Teman>();
    while (dr.Read())
    {
        //Teman
        string username1 = dr.GetValue(0).ToString();
        string username2 = dr.GetValue(1).ToString();
        DateTime tglBerteman = DateTime.Parse(dr.GetValue(2).ToString());
        string status = dr.GetValue(3).ToString();

        string friendUSN = username1;
        if (friendUSN == username)
        {
            friendUSN = username2;
        }

        //Create Teman(buffer)
        Teman newTeman = new Teman(friendUSN, tglBerteman, status);

        //Add to list teman
        listTeman.Add(newTeman);
    }
    return listTeman;
}

```

- *Method* ini digunakan untuk mengakses *request pertemanan* terkait current_user dari *database*, yang nantinya akan ditampilkan di *interface*. Data-data akan diambil dari

tabel teman yang memiliki status selain “Berteman”. Data-data tersebut akan disimpan di dalam *variable* untuk dibuat *object* nya melalui *constructor* yang ada di *class Teman*. *Object-object* yang telah berisi data-data tersebut akan dimasukkan ke dalam *list* dan dikembalikan sebagai hasil.

2.16 Class DAO_Komen

```
1 reference
public static void Insert_Komen(Komen newKomen, int konten_id)
{
    string command = "INSERT INTO `pameryuk`.`komen` (`id`, `Komentar`, `"
        +"tgL`, `username`, `Konten_id`) " +
        "VALUES ('"+Get_NewComment_Id().ToString()+"', '"
        +newKomen.Komentar+"', '"
        +newKomen.Tgl.ToString("yyyy-MM-dd HH:mm:ss")+"', '"
        +newKomen.User+"', '"+konten_id+"');";
    KoneksiDatabase.DatabaseDMLCommand(command);
}
```

- Method ini digunakan untuk menambahkan setiap komentar baru ke dalam database tanpa melakukan query berulang

```
3 references.
public class DAO_Komen
{
    2 references
    public static List<Komen> Select_Komen(int konten_id)
    {
        string command = "select * from komen where konten_id = '"+konten_id+"' ;";
        MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(command);
        List<Komen> listKomen = new List<Komen>();
        Komen kamen;
        while (dr.Read())
        {
            //User
            int id = int.Parse(dr.GetValue(0).ToString());
            string komentar = dr.GetValue(1).ToString();
            DateTime tglUpload = DateTime.Parse(dr.GetValue(2).ToString());
            string username = dr.GetValue(3).ToString();

            kamen = new Komen(id, komentar, tglUpload, username);
            listKomen.Add(kamen);
        }
        return listKomen;
    }
}
```

- *Method* ini digunakan untuk mengakses komen-komen pada konten tertentu. 4 data dari konten yang terdiri dari id, komentar, tanggal upload, dan username akan disimpan ke dalam *variable* terlebih dahulu, untuk dikirimkan ke *constructor* yang ada di *class Komen*. *Object-object* yang telah dibuat menggunakan *constructor* akan dimasukkan ke dalam *list* dan dikembalikan sebagai hasil.

```
1 reference
private static int Get_NewComment_Id()
{
    string command = "select id from komen order by id desc limit 1;";
    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(command);
    int result=-1;
    if (dr.Read())
    {
        //User
        result = int.Parse(dr.GetValue(0).ToString());
        result++;
    }
    return result;
}
```

- Dengan adanya method ini, ID dari komentar akan selalu unik tanpa ada yang sama, dan id akan tercatat secara berurut dan sequential. Hal ini akan memastikan bahwa komen tidak ada yang terduplicasi.

2.17 Class DAO_Konten

```
public static void Insert_Konten(Konten newKonten, string username)

{
    int konten_id=Get_NewKonten_Id();

    string command = "INSERT INTO `pameryuk`.`konten` (`id`, `caption`, `fotoDiri`, `video`, `tglUpload`, `username`) VALUES (" + konten_id + ", " + newKonten.Caption + ", " + newKonten.Foto + ", " + newKonten.Video + ", " + newKonten.TglUpload.ToString("yyyy-MM-dd HH:mm:ss") + ", " + username + ");";

    KoneksiDatabase.DatabaseDMLCommand(command);
}
```

- *Method* ini digunakan untuk mengeksekusi perintah untuk menambahkan konten baru. Untuk method ini, kita dapat menambahkan 6 dari 8 *variable* yang dimiliki *class* Konten. *Like* dan *comment* belum dapat diisi sewaktu penambahan konten baru.

```
6 references
public static List<Konten> Select_ListKonten(string username)
{
    string perintah = "SELECT * from konten where username = '" + username + "' ;";

    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(perintah);
    List<Konten> listKonten = new List<Konten>();
    Konten konten;
    while (dr.Read())
    {
        //Konten
        int id = int.Parse(dr.GetValue(0).ToString());
        string caption = dr.GetValue(1).ToString();
        string foto = dr.GetValue(2).ToString();
        string video = dr.GetValue(3).ToString();
        DateTime tglUpload = DateTime.Parse(dr.GetValue(4).ToString());

        //Create Konten
        konten = new Konten(id, caption, foto, video, tglUpload);
        listKonten.Add(konten);
    }
    return listKonten;
}
```

- *Method* ini digunakan untuk mengakses konten-konten yang dibuat oleh *username* tertentu. 5 data dari konten yang terdiri dari id, caption, foto, video, dan tanggalUpload akan disimpan ke dalam *variable* terlebih dahulu, untuk dikirimkan ke *constructor* yang ada di *class* Konten. *Object-object* yang telah dibuat menggunakan *constructor* akan dimasukkan ke dalam *list* dan dikembalikan sebagai hasil.

```
3 references
public static Konten Select_Konten(int konten_id)
{
    string perintah = "SELECT * from konten where id = '" + konten_id + "'";

    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(perintah);
    List<Konten> listKonten = new List<Konten>();
    Konten konten;
    if (dr.Read())
    {
        //Konten
        int id = int.Parse(dr.GetValue(0).ToString());
        string caption = dr.GetValue(1).ToString();
        string foto = dr.GetValue(2).ToString();
        string video = dr.GetValue(3).ToString();
        DateTime tglUpload = DateTime.Parse(dr.GetValue(4).ToString());

        //Create Konten
        konten = new Konten(id, caption, foto, video, tglUpload);
        return konten;
    }
    else return null;
}
```

- *Method* ini menggunakan konsep yang sama seperti *method* sebelumnya, yang membedakan ialah konten yang diakses akan berdasarkan id dari konten. Karena id untuk tiap konten berbeda-beda, maka hasil yang dikembalikan hanya berupa 1 *object* Konten.

```
2 references
public static int Get_NewKonten_Id()
{
    string command = "select id from konten order by id desc limit 1";
    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(command);
    int newId=0;
    if(dr.Read())
    {
        newId = int.Parse(dr.GetValue(0).ToString());
    }
    return (newId + 1);
}
```

- Method `Get_NewKonten_Id` digunakan untuk menentukan *id* untuk konten yang akan ditambahkan. Cara kerja method ini adalah dengan mengakses *id* terbesar yang ada di dalam tabel konten. Dengan demikian, *id* untuk konten baru didapatkan dengan menambahkan nilai 1 pada *id* terbesar.

2.18 Class DAO_KisahHidup

```
public static List<KisahHidup> Select_ListKisahHidup(string username)
{
    string command;
    command = "select * from KisahHidup kh inner join organisasi o ON o.ID = kh.Organisasi_id inner join Kota K on o.Kota_Id = K.id where kh.username ='" + username + "'";

    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(command);
    List<KisahHidup> listData = new List<KisahHidup>();

    while (dr.Read() == true)
    {
        //Taken from database
        //KisahHidup
        int organisasi_ID_fk = int.Parse(dr.GetValue(0).ToString());
        string username_fk = dr.GetValue(1).ToString();
        int thawal = int.Parse(dr.GetValue(2).ToString());
        int thakhir = int.Parse(dr.GetValue(3).ToString());
        string deskripsi = dr.GetValue(4).ToString();

        //Organisasi
        int id_organisasi = int.Parse(dr.GetValue(5).ToString());
        string nama_organisasi = dr.GetValue(6).ToString();
        int kota_id_fk = int.Parse(dr.GetValue(7).ToString());

        //Kota
        int kota_id = int.Parse(dr.GetValue(8).ToString());
        string nama_kota = dr.GetValue(9).ToString();

        //Create Kota
        Kota newKota = new Kota(kota_id, nama_kota);

        //Create Organisasi
        Organisasi newOrganisasi = new Organisasi(id_organisasi, nama_organisasi, newKota);

        //Create Kisah Hidup
        KisahHidup newKisah = new KisahHidup(newOrganisasi, thawal, thakhir, deskripsi);

        //Tambahkan ke list
        listData.Add(newKisah);
    }
    //Kirim kembali list ke pemanggilnya
    return listData;
}
```

- Method ini mengambil semua kisah hidup pengguna berdasarkan username, termasuk informasi organisasi dan kota. Tabel sumber tersusun atas tabel kisah hidup yang digabungkan dengan tabel organisasi, lalu digabungkan dengan tabel kota. Data terstruktur (Kota, organisasi dan kisah hidup) ini dapat digunakan untuk ditampilkan di UI, dianalisis, atau diproses lebih lanjut.

```
I reference
public static void Insert_KisahHidup(KisahHidup newKisah, User current_user)
{
    string command = "INSERT INTO `pameryuk`.`kisahhidup` " +
        "(`organisasi_id`, `username`, `thawal`, `thakhir`, `deskripsi`) " +
        "VALUES ('" + newKisah.Organisasi.Id + "', '" + current_user.Username +
        "', '" + newKisah.Thawal + "', '" + newKisah.Thakhir + "', '" +
        newKisah.Deskripsi+ "')";

    KoneksiDatabase.DatabaseDMLCommand(command);
}
```

- Method ini mempermudah penambahan data pada kisah hidup ke *database* secara otomatis tanpa harus melakukan query berulang.

2.19 Class DAO_Tag

```
1 reference
public static List<User> Select_Tag(int konten_id)
{
    string command = "select * from tag inner join user on tag.username = user.username inner join kota on user.kota_id = kota.id where konten_id = '" + konten_id + "'";

    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(command);
    List<User> listAkun = new List<User>();
    User user;
    while (dr.Read())
    {
        //User
        string username = dr.GetValue(0).ToString();
        string namalengkap = dr.GetValue(2).ToString();
        DateTime tglahir = DateTime.Parse(dr.GetValue(3).ToString());
        string noKTP = dr.GetValue(4).ToString();
        string fotoDiri = dr.GetValue(5).ToString(); //Still confuse with this image data format
        string fotoProfil = dr.GetValue(6).ToString();
        string email = dr.GetValue(7).ToString();
        int kota_id_fk = int.Parse(dr.GetValue(8).ToString());

        //Kota
        int kota_id = int.Parse(dr.GetValue(9).ToString());
        string nama = dr.GetValue(10).ToString();

        //Create Kota
        Kota newKota = new Kota(kota_id, nama);

        //Create User
        user = new User(username, namalengkap, tglahir, noKTP, fotoDiri, fotoProfil, email, newKota);
        listAkun.Add(user);
    }
    return listAkun;
}
```

- Method ini mengambil semua *tag* pada konten tertentu. Tabel sumber tersusun atas tabel tag yang digabungkan dengan tabel user, lalu digabungkan dengan tabel kota. Data-data terkait kota akan dikirimkan ke *constructor* yang ada di *class* Kota. Selanjutnya, *object* Kota dan data-data lainnya akan disimpan ke dalam *variable* terlebih dahulu, untuk dikirimkan ke *constructor* yang ada di *class* User. *Object-object* yang telah dibuat menggunakan *constructor* akan dimasukkan ke dalam *list* dan dikembalikan sebagai hasil.

```
1 reference
public static void Insert_Tag(int konten_id, string username)
{
    string command = "INSERT INTO `pameryuk`.`tag` (`Konten_id`, `username`) VALUES ('" + konten_id + "', '" + username + "');";
    KoneksiDatabase.DatabaseDMLCommand(command);
}
```

- *Method* Insert_Tag digunakan untuk menambahkan *tag* baru ke *database*. *Tag* sendiri tersusun dari 2 *variable* (id konten dan username), dimana ke-2 data tersebut dapat ditambahkan melalui *method* ini tanpa harus melakukan query berulang.

3. Kreasi Fitur

- **Tujuan Fitur :** Sehubungan dengan aplikasi PamerYuk yang termasuk ke dalam aplikasi pertemanan, tentunya diperlukan fitur yang dapat membuat hubungan pertemanan antara *user* menjadi lebih erat. Oleh karena itu, kami hendak menambahkan 2 fitur ini :

1. Fitur Chat

Pada fitur ini, *user* dapat saling mengirim pesan dengan *user* lain yang telah menjadi temannya. Pembuatan fitur ini ditujukan agar *user* dapat menjalin hubungan yang lebih privat dengan beberapa *user* tertentu. Diharapkan seiring waktu, hubungan pertemanan diantara *user-user* yang menggunakan aplikasi PamerYuk dapat menjadi lebih erat seiring berkomunikasi.

2. Fitur Like

Dengan adanya fitur ini, *user* dapat menyukai konten yang diunggah oleh *user* lain. Melalui fitur ini, *user* dapat menemukan kesenangan baru sekaligus dapat menemukan teman dengan hobi / peminatan yang sama seiring waktu.

- **Cara Kerja**

- **Coding :**

1. **Class Service :**

```
2 references
public bool Check_Like(int konten_id)
{
    return DAO_Like.CheckLike(konten_id, this.Current_user.Username);
}

1 reference
public Konten Tambah_Like(int konten_id)
{
    DAO_Like.Insert_Like(konten_id, this.Current_user.Username);
    return DAO_Konten.Select_Konten(konten_id);
}

1 reference
public Konten Delete_Like(int konten_id)
{
    DAO_Like.DELETE_Like(konten_id, this.Current_user.Username);
    return DAO_Konten.Select_Konten(konten_id);
}
#endifregion
```

- *Method Check_Like* digunakan untuk menghitung berapa banyak *like* yang ada pada konten. Proses perhitungan dilakukan dengan memanggil *method* di *DAO_Like* dengan parameter id konten dan username dari *current_user*.

- *Method* Tambah_Like digunakan untuk menambahkan banyak *like* yang ada pada konten. Proses penambahan dilakukan dengan memanggil *method* di DAO_Like dengan parameter id konten. Setelah itu, dilakukan proses *refresh* pada konten yang bersangkutan dengan mengambil data terbaru dari *database*.
- *Method* Delete_Like digunakan untuk menghapus *like* yang sebelumnya telah dilakukan oleh *user* tertentu. Proses penghapusan dilakukan dengan memanggil *method* di DAO_Like dengan parameter id konten dan *username* yang bersangkutan. Setelah itu, dilakukan proses *refresh* pada konten yang bersangkutan dengan mengambil data terbaru dari *database*.

```
#region CHAT

1 reference
public List<Chat> Buka_Chat(string username)
{
    return DAO_Chat.Select_Chat(username, this.Current_user.Username);
}

1 reference
public void Kirim_Chat(Chat chat)
{
    DAO_Chat.Insert_Chat(chat);
}
#endregion
```

- *Method* Buka_Chat digunakan untuk membuka *chat*, baik itu *chat* yang dikirimkan oleh *user* maupun yang diterima oleh *user*. Proses buka *chat* dilakukan dengan memanggil *method* di DAO_Chat dengan parameter *username* dari teman dan *username* dari *current_user*.
- *Method* Kirim_Chat digunakan untuk menambahkan data *chat* baru ke dalam *database*. Proses penambahan *chat* ini dilakukan dengan memanggil *method* di DAO_Chat dengan parameter *chat* yang dikirimkan dari *form*.

2. Class Entity Chat :

```
12 references
public class Chat
{
    private int id;
    private string pesan;
    private string pengirim;
    private string penerima;
    private DateTime tglTerkirim;

    1 reference
    public Chat(int id, string pesan, string pengirim, string penerima, DateTime tglTerkirim)
    {
        this.Id = id;
        this.Pesan = pesan;
        this.Pengirim = pengirim;
        this.Penerima = penerima;
        this.TglTerkirim = tglTerkirim;
    }

    1 reference
    public Chat(string pesan, string pengirim, string penerima) //for new chat
    {
        this.Pesan = pesan;
        this.Pengirim = pengirim;
        this.Penerima = penerima;
    }
}
```

- Bagian *Variables*:

1. private int id;
 - *Variable* ini digunakan untuk menyimpan kode unik pada setiap *chat*.
2. private string pesan;
 - *Variable* ini digunakan untuk menyimpan teks ataupun pesan yang dikirimkan.
3. private string pengirim;
 - *Variable* ini digunakan untuk menyimpan data mengenai siapa yang mengirim *chat* tersebut.
4. private string penerima;
 - *Variable* ini digunakan untuk menyimpan data mengenai siapa yang menerima *chat* tersebut.
5. private DateTime tglTerkirim;
 - *Variable* ini digunakan untuk menyimpan tanggal ketika *chat* tersebut dikirimkan.

- Bagian *Constructors*:

1. public Chat(int id, string pesan, string pengirim, string penerima, DateTime tglTerkirim)

- *Constructor* ini dipanggil ketika terjadi proses *input* data chat ke *database* untuk yang pertama kalinya / proses pengambilan data chat dari *database*.
2. public Chat(string pesan, string pengirim, string penerima)
 - Berbeda dengan yang sebelumnya, *constructor* ini akan dipanggil ketika terjadi proses penambahan data ke tabel chat yang sudah terisi. *Variable id* tidak perlu disertakan karena tersedia fitur *auto increment* yang akan secara otomatis mengisi bagian *id*.

```

1 reference
public int Id { get => id; set => id = value; }
4 references
public string Pesan { get => pesan; set => pesan = value; }
4 references
public string Pengirim { get => pengirim; set => pengirim = value; }
3 references
public string Penerima { get => penerima; set => penerima = value; }
2 references
public DateTime TglTerkirim { get => tglTerkirim; set => tglTerkirim = value; }

0 references
public override string ToString()
{
    return "[" + this.TglTerkirim.ToString() + "] " + this.Pengirim + " : " + this.Pesan + "\n";
}

```

- Bagian *Property*
 1. public int Id;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable id*, yang memuat *index* / kode unik pada *setiap chat*.
 2. public string Pesan;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable pesan*, yang memuat teks ataupun pesan yang dikirimkan.
 3. public string Pengirim;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable pengirim*, yang memuat data mengenai siapa yang mengirim *chat* tersebut.
 4. public string Penerima;
 - *Property* ini digunakan untuk mengakses / mengubah data dari *variable penerima*, yang memuat data mengenai siapa yang menerima *chat* tersebut.

5. public DateTime TglTerkirim;

- *Property* ini digunakan untuk mengakses / mengubah data dari *variable* tglTerkirim, yang memuat data mengenai tanggal ketika *chat* tersebut dikirimkan.

• Bagian *Method*

- *Method* ini akan dipanggil ketika terdapat *object Chat* yang akan diubah menjadi *string*. Dengan adanya *method* ini, keambiguan dapat dicegah dan data yang akan dikembalikan adalah tanggal pengiriman chat, *user* yang menjadi pengirim, dan pesan yang dikirimkan.

3. Class Entity *DAO_Like* dan *DAO_Chat* :

```
4 references
public class DAO_Like
{
    1 reference
    public static int Select_LikeCount(int konten_id)
    {
        string command = "SELECT count(username) FROM pameryuk.'like' where konten_id = '" + konten_id + "'";
        int result = 0;
        MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(command);
        if (dr.Read())
        {
            result = int.Parse(dr.GetValue(0).ToString());
        }
        return result;
    }
    1 reference
    public static bool CheckLike(int konten_id, string username)
    {
        string command = "SELECT count(username) FROM pameryuk.'like' where konten_id = '" + konten_id + "' and username = '" + username + "'";
        int result = 0;
        MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(command);
        if (dr.Read())
        {
            result = int.Parse(dr.GetValue(0).ToString());
        }
        if(result != 0)
        {
            return true;
        }
        else return false;
    }
}
```

- *Method Select_LikeCount* digunakan untuk menghitung berapa banyak like yang ada pada suatu konten. Sumber tabel yang digunakan adalah tabel like dengan menggunakan *command Count* untuk menghitung banyaknya username. Hasil perhitungan ini akan dikembalikan sebagai hasil.
- *Method CheckLike* juga menggunakan konsep yang sama dengan method *Select_LikeCount*. Yang menjadi pembeda ialah method ini digunakan untuk menentukan *action* yang dilakukan oleh *user*. Apabila hasilnya true, maka *user* sedang melakukan like, begitu juga sebaliknya.

```

public static void Insert_Like(int konten_id, string username)

{
    string command = "INSERT INTO `pameryuk`.`like` (`Konten_id`, `username`) VALUES
    (" + konten_id + "", "" + username + "");;

    KoneksiDatabase.DatabaseDMLCommand(command);
}

public static void DELETE_Like(int konten_id, string username)

{
    string command = "DELETE FROM `pameryuk`.`like` where konten_id = " + konten_id +
    " and username = " + username + ";;

    KoneksiDatabase.DatabaseDMLCommand(command);
}

```

- *Method Insert_Like* digunakan untuk menambahkan data like baru ke *database*. *Class Like* sendiri tersusun dari 2 *variable* (id konten dan username), dimana ke-2 data tersebut dapat ditambahkan melalui *method* ini.
- *Method Delete_Like* digunakan untuk menghapus like yang diberikan oleh user tertentu pada konten tertentu. Id konten dan username akan dikirimkan sebagai parameter untuk nantinya dijadikan acuan.

```

public class DAO_Chat

{
    public static List<Chat> Select_Chat(string friend, string user)

    {
        //usn2 is current user

        string perintah = "SELECT * FROM chat WHERE pengirim = " + friend + " and
        penerima = " + user + " UNION SELECT * FROM chat WHERE pengirim = " + user + "
        and penerima = " + friend + " order by id asc;";
    }
}

```

```

MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(perintah);

List<Chat> listChat = new List<Chat>();

while (dr.Read())
{
    int id = int.Parse(dr.GetValue(0).ToString());

    string pesan = dr.GetValue(1).ToString();

    DateTime tglTerkirim = DateTime.Parse(dr.GetValue(2).ToString());

    string pengirim = dr.GetValue(3).ToString();

    string penerima = dr.GetValue(4).ToString();

    Chat newChat = new Chat(id,pesan,pengirim,penerima,tglTerkirim);

    listChat.Add(newChat);
}

return listChat;
}

```

- *Method* ini digunakan untuk mengakses chat terkait current_user dari *database*, yang nantinya akan ditampilkan di *interface*. Data-data akan diambil dari tabel chat di *database*. Data-data tersebut akan disimpan di dalam *variable* untuk dibuat *object* nya melalui *constructor* yang ada di *class Chat*. *Object-object* yang telah berisi data-data tersebut akan dimasukkan ke dalam *list* dan dikembalikan sebagai hasil.

```

public static void Insert_Chat(Chat chat)
{
    string command = "INSERT INTO `pameryuk`.`chat` (`id`, `pesan`, `tglTerkirim`, `pengirim`, `penerima`) VALUES (" + Get_NewChat_Id() + "", "" + chat.Pesan + "", "" + DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss") + "", "" + chat.Pengirim + "", "" + chat.Penerima + "");";

```

```
KoneksiDatabase.DatabaseDMLCommand(command);  
}
```

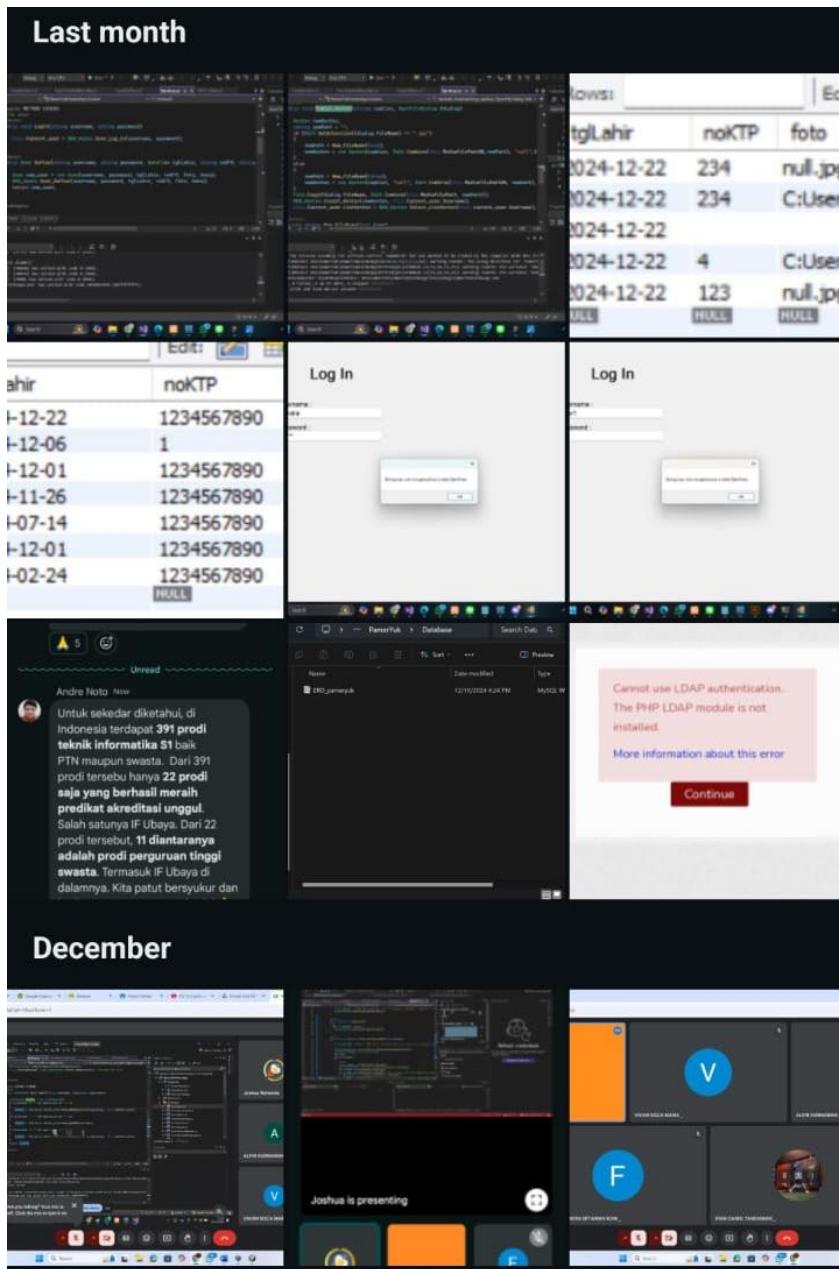
- *Method Insert_Chat* digunakan untuk menambahkan chat baru ke *database*. *Class Chat* sendiri tersusun dari 5 *variable*, dimana ke-5 data tersebut dapat ditambahkan melalui *method* ini.

```
private static int Get_NewChat_Id()  
{  
    int result=0;//for sementara  
  
    string perintah = "SELECT id FROM chat ORDER BY id DESC LIMIT 1;";  
  
    MySqlDataReader dr = KoneksiDatabase.DatabaseQueryCommand(perintah);  
  
    if (dr.Read())  
    {  
        result = int.Parse(dr.GetValue(0).ToString());  
    }  
  
    return (result+1);  
}
```

- Dengan adanya method ini, ID dari chat akan selalu unik tanpa ada yang sama, dan id akan tercatat secara berurut dan sequential. Hal ini akan memastikan bahwa komen tidak ada yang terduplicasi.

4. Dokumentasi Kerja Kelompok

A. Dokumentasi Kerja Kelompok Pertama



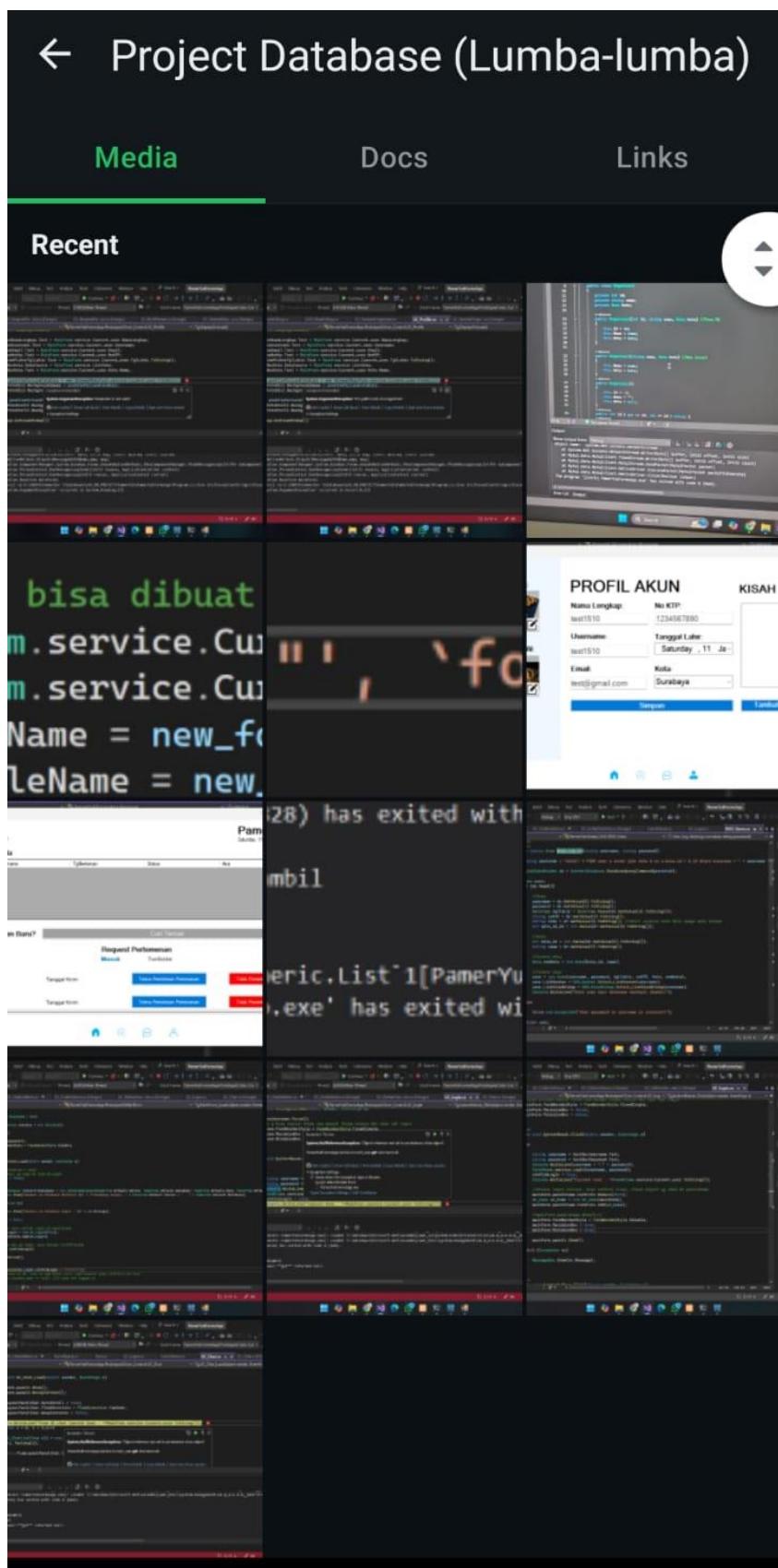
Dokumentasi Kerja Kelompok Pertama :

Jadwal : Desember-Januari

Tempat : Online Whatsapp + Github

Hasil Kerja Kelompok : Membahas class dan data di database

B. Dokumentasi Kerja Kelompok Kedua



Dokumentasi Kerja Kelompok Kedua :

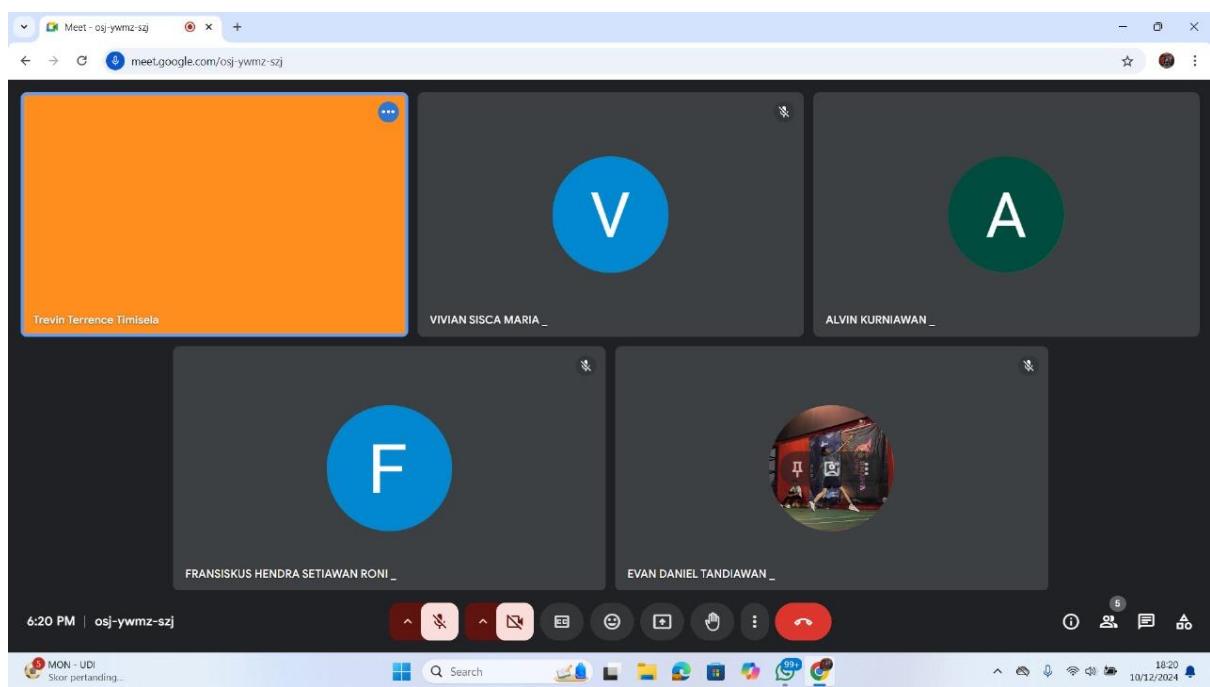
Jadwal : Januari Awal

Tempat : Online Whatsapp + Github

Hasil Kerja Kelompok : Menyelesaikan class, UI design, dan menghubungkan form

5. Dokumentasi Konsultasi Kakak Asdos

A. Konsultasi Pertama

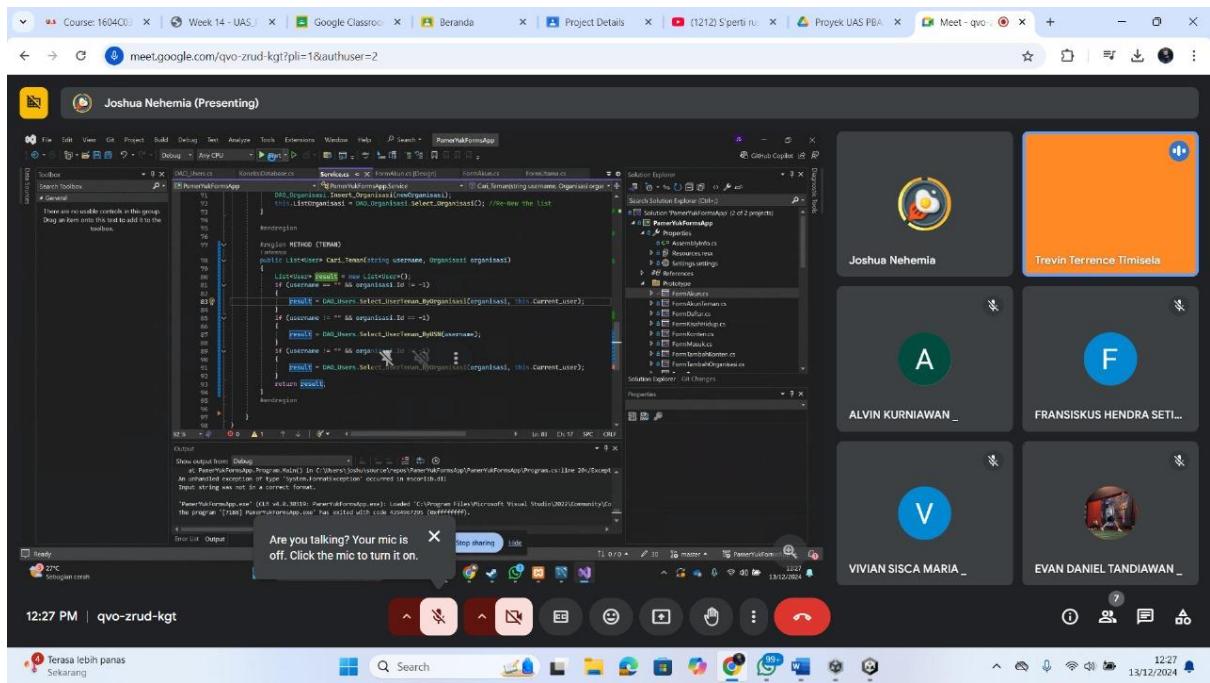


Jadwal : 10/12/2024

Tempat : Online Gmeet

Hasil Konsultasi : Membahas beberapa class

B. Konsultasi Kedua



Jadwal : 13/12/2024

Tempat : Online Gmeet

Hasil Konsultasi : Melakukan pengecekan pada class yang sudah selesai