

**MODUL PEMBUATAN APLIKASI  
PENDAFTARAN KKP & KKN  
MENGGUNAKAN APLIKASI NETBEANS**



**GLOBAL INSTITUTE  
OF TECHNOLOGY & BUSINESS**  
*Think Smartly & Globally*

**Disusun Oleh:**

<b>Nama Mahasiswa</b>	<b>NIM</b>
Ilham Aditya Putra Budi	1122140049
Mochamed Fadhlhan Tuhairi	1122140146
Eka Putu Miharja	1122140013
Muhammad Zaenal Abby S.	1122140009

**PROGRAM STUDI TEKNIK INFORMATIKA  
KONSENTRASI SOFTWARE ENGINEERING  
INSTITUT TEKNOLOGI DAN BISNIS BINA SARANA GLOBAL  
TANGERANG  
2024**

**MODUL PEMBUATAN APLIKASI  
PENDAFTARAN KKP & KKN  
MENGGUNAKAN APLIKASI NETBEANS**

Dosen Pembimbing:  
Rahmat Tullah, M.Kom  
Agustinus Sirumapea, S.P., M.Kom.  
Detin Sofia, M.Kom.



**GLOBAL INSTITUTE  
OF TECHNOLOGY & BUSINESS**  
*Think Smartly & Globally*

**Disusun Oleh:**

<b>Nama Mahasiswa</b>	<b>NIM</b>
Ilham Aditya Putra Budi	1122140049
Mochamed Fadhlwan Tuhairi	1122140146
Eka Putu Miharja	1122140013
Muhammad Zaenal Abby S.	1122140009

**PROGRAM STUDI TEKNIK INFORMATIKA  
KONSENTRASI SOFTWARE ENGINEERING  
INSTITUT TEKNOLOGI DAN BISNIS BINA SARANA GLOBAL  
TANGERANG  
2024**

## **KATA PENGANTAR**

Segala puji syukur kami panjatkan kepada Tuhan Yang Maha Esa, Allah SWT, atas rahmat-Nya sehingga penulisan modul Mata Kuliah Project II dapat terselesaikan dengan baik. Modul ini disusun untuk memenuhi kebutuhan mahasiswa dalam mata kuliah Project II dengan menggunakan bahasa pemrograman Java. Kami menyusun modul ini agar mudah dipahami oleh mahasiswa, dengan merangkum materi dari pertemuan awal hingga pertemuan akhir. Modul ini berfokus pada proyek aplikasi pendaftaran KKP dan KKN yang mencakup Front End dan Back End. Tujuan dari modul ini adalah untuk mendukung pembelajaran mahasiswa dalam membuat aplikasi pendaftaran KKP dan KKN yang dinamis. Dengan adanya modul ini, kami berharap mahasiswa dapat lebih mudah dalam mengembangkan aplikasi pendaftaran menggunakan bahasa pemrograman Java. Selain itu, program ini dirancang untuk memudahkan siapa saja yang ingin menggunakan atau mengembangkannya lebih lanjut. Modul ini disediakan dengan sumber kode open source, sehingga memungkinkan pengembangan lebih lanjut dengan menu dan tampilan yang berbeda sesuai kebutuhan pengguna. Modul ini juga diharapkan dapat berguna bagi siapa saja yang ingin mempelajari dan mengembangkannya.

Tangerang,10 Agustus 2024

Penulis

## DAFTAR ISI

KATA PENGANTAR.....	2
DAFTAR ISI .....	3
BAB I PEMBAHASAN .....	4
1.1 Apa Itu Apache NetBeans .....	4
1.2 Bagamana Cara Melakukan Instalasi Pada NeatBeans .....	5
BAB II.....	7
MERANCANG & MEMBUAT DATABASE PADA APLIKASI PENDAFTARAN KKP & KKN.....	7
2.1 Membuat Database pada Netbeans .....	7
2.1.1 UseCase Diagram .....	7
2.1.2 Activity Diagram .....	9
2.1.3 Sequence Diagram.....	27
2.1.4 Class Diagram.....	41
2.1.5. Membuat Database .....	42
BAB III MEMBUAT VIEW .....	44
3.1 Konfigurasi pada Database pada Apache Netbeans .....	44
3.2 Buat Jframe Menu Utama .....	45
3.3 Frame login .....	59
3.4 Frame Pengguna.....	62
3.5 Frame mahasiswa .....	70
3.6 Frame dosen .....	78
3.7 Frame Tempat kkn.....	85
3.8 Frame tempat kkp.....	97
3.9 Frame daftar kkn .....	113
3.11 Frame daftar kkp .....	124
3.12 Evaluasi KKN .....	135
3.13 Evaluasi KKP .....	142
BAB IV PANDUAN PENGGUNAAN .....	149
4.1 View Halaman Utama Admin .....	149
4.2 Halaman Utama Dosen.....	151
4.3 Halaman Utama Mahasiswa.....	151

## BAB I

### PEMBAHASAN

#### 1.1 Apa Itu Apache NetBeans

Penjelasan pada modul ini dimaksudkan untuk memperkenalkan mahasiswa atau pengguna pada Netbeans dan prinsip dasar arsitektur MVC. Ini akan menunjukkan kepada pengguna bagaimana aplikasi Netbeans dasar dibangun dengan cara selangkah demi selangkah. dalam tutorial ini, pengguna akan membuat aplikasi berita dasar. Anda akan mulai dengan menulis kode yang dapat memuat halaman statis. Selanjutnya, pengguna akan membuat rubrik berita yang membaca item berita dari database. Terakhir, pengguna akan menambahkan formulir untuk membuat item berita di database.

Tutorial ini terutama akan berfokus pada:

1. Model-View-Controller basics
2. Routing basics
3. Form validation
4. Performing basic database queries using Query Builder

Kami akan menjelaskan mengenai MVC (*Model, View, Controller*) pada Netbeans adalah sebuah pola desain (*design pattern*) arsitektur pengembangan aplikasi yang memisahkan dan mengelompokan beberapa kode sesuai dengan fungsinya.<sup>1</sup>

MVC terbagi bagi aplikasi ke dalam tiga bagian fungsional: model, view, dan controller. Berikut adalah penjelasan dari MVC

1. **Model** adalah kode-kode untuk model bisnis dan data. biasanya berhubungan langsung dengan database untuk memanipulasi data (insert, update, delete, search), menangani validasi dari bagian controller, namun tidak dapat berhubungan langsung dengan bagian view.
2. **View** merupakan bagian yang menangani *presentation logic*. berisi kode-kode untuk tampilan.

3. **Controller** merupakan bagian yang mengatur hubungan antara bagian *model* dan bagian *view*, controller berfungsi untuk menerima request dan data dari user kemudian menentukan apa yang akan diproses oleh aplikasi.

## 1.2 Bagaimana Cara Melakukan Instalasi Pada NetBeans

1. Pastikan bahwa koneksi internet anda stabil dan lancar agar tidak akan mengganggu proses download installer netbeans.
2. Anda harus mengerti jenis windows pada perangkat anda, misalkan menggunakan  
32-bit (x86) atau 64-bit (x64)
3. Anda bisa mendapatkan installer netbeans dengan mencari di google, masukkan keyword **download installer netbeans**
4. Pilihan lain juga anda dapat mendownload pada situs  
<https://www.oracle.com/technetwork/java/javase/downloads/jdknetbeans-jsp-142931.html>.
5. Kemudian anda akan langsung dihubungkan dengan website **apache netbeans**, pilih menu **download** pada bagian pojok kanan atas
6. Kemudian anda pilih jenis netbeans yang akan anda install. Pilih **Apache Netbeans 11 Feature Update 1 (NB 11.1)**, klik **Download**
7. Tunggu hingga proses download installer selesai. Lama waktu download akan bergantung pada kecepatan akses internet yang digunakan pada saat itu, sehingga proses download bisa cepat ataupun lambat.
8. Cari file download netbeans tersebut, dan klik 2 kali pada file installer netbeans.
9. Kemudian akan muncul proses penginstallan netbeans yang diconfigurasikan dulu secara otomatis oleh installer netbeans tersebut. Hal ini untuk mendeteksi apakah perangkat anda dapat digunakan untuk menjalankan program netbeans atau tidak.  
Tunggu proses ini hingga selesai
10. Setelah itu, akan muncul tampilan install **Java SE Development Kit**, klik **Next**

11. Anda akan diminta untuk menyetujui **lisence agreement**, pilih pilihan **I accept the terus ..., Klik Next**
12. Kemudian sesuaikan lokasi penyimpanan dari JDK dan Netbeans IDE pada perangkat anda. Bebas menggunakan direktori yang anda inginkan, setelah sudah ditetapkan direktorinya, klik **Next**.
13. Kemudian akan muncul ringkasan dari apa saja yang akan diinstall, klik **Install**. Pada tahap ini anda dapat mereview kembali apakah yang akan diinstall sudah termasuk Netbeans, JDK/JRE, ketika semua hal tersebut sudah dapat ditemukan, maka anda dapat mengklik **Install**.
14. Tunggu hingga proses installasi selesai dan muncul pesan bahwa proses installasi berhasil, kemudian klik **finish**.

Proses installasi netbeans yang dijelaskan pada tutorial ini berfokus pada installasi 1 paket. Sehingga anda tidak perlu bingung dan khawatir akan susah dan rumitnya proses menginstall secara terpisah. Karena dengan satu kali proses installasi, semua package seperti **JDK/JRE**, **Netbeans** akan terinstall secara bersamaan dan hanya membutuhkan satu kali proses installasi.

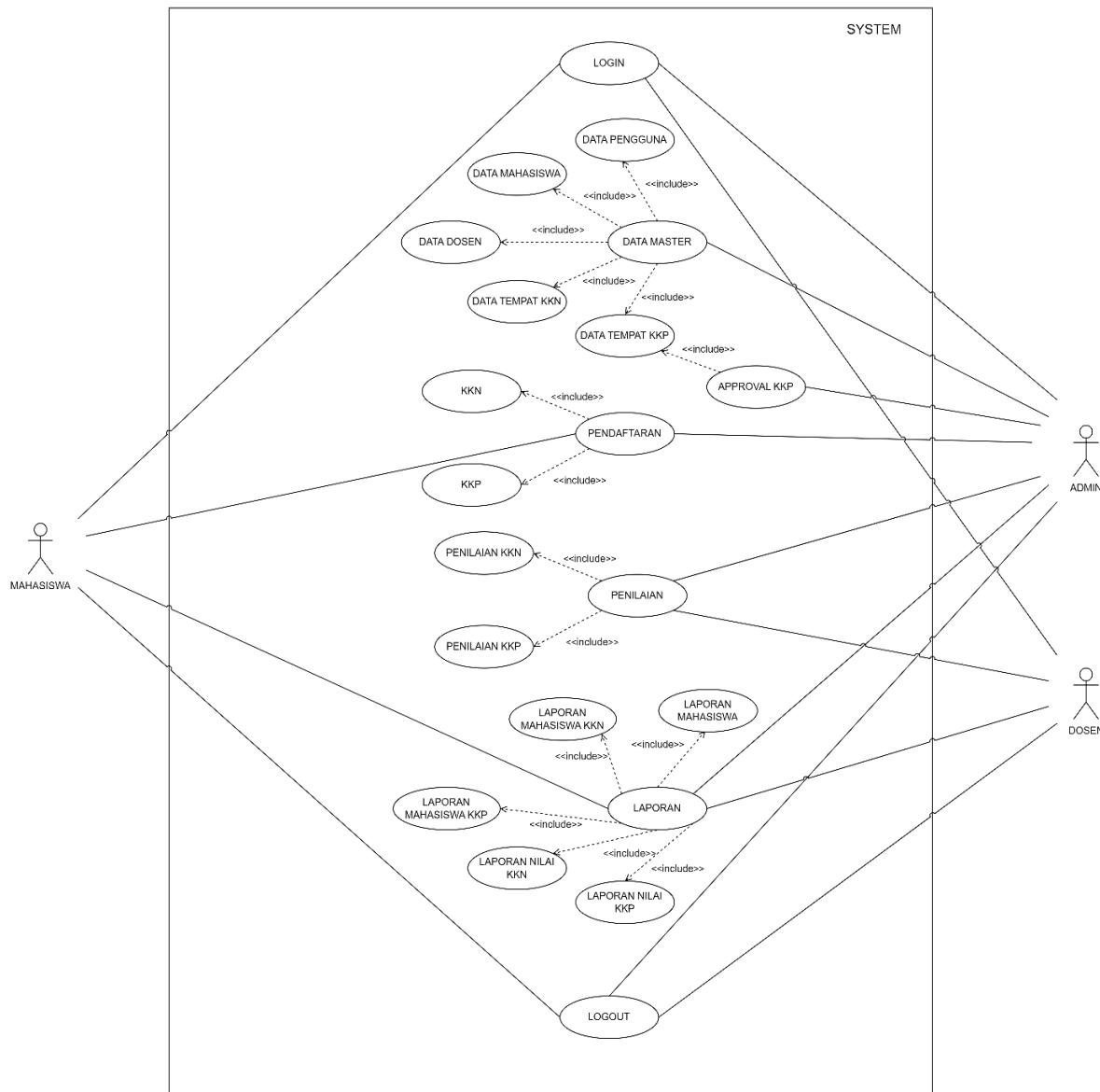
## BAB II

### MERANCANG & MEMBUAT DATABASE PADA APLIKASI PENDAFTARAN KKP & KKN

#### 2.1 Membuat Database pada Netbeans

##### 2.1.1 UseCase Diagram

Program yang ingin dibuat menggunakan 3 aktor yang terdiri dari Admin, Mahasiswa, dan Dosen. Dengan memiliki peran dan fungsinya masing-masing.



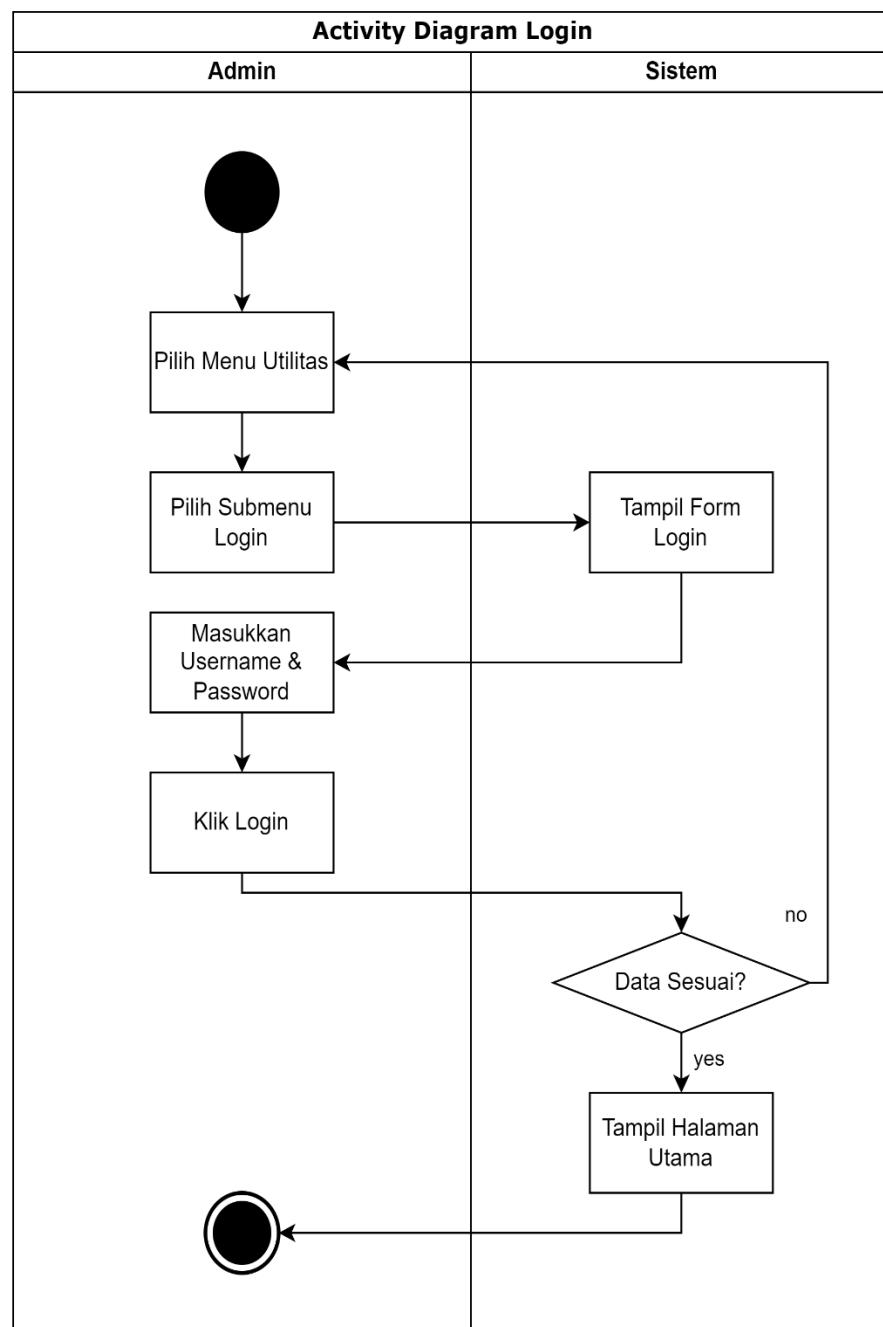


### 2.1.2 Activity Diagram

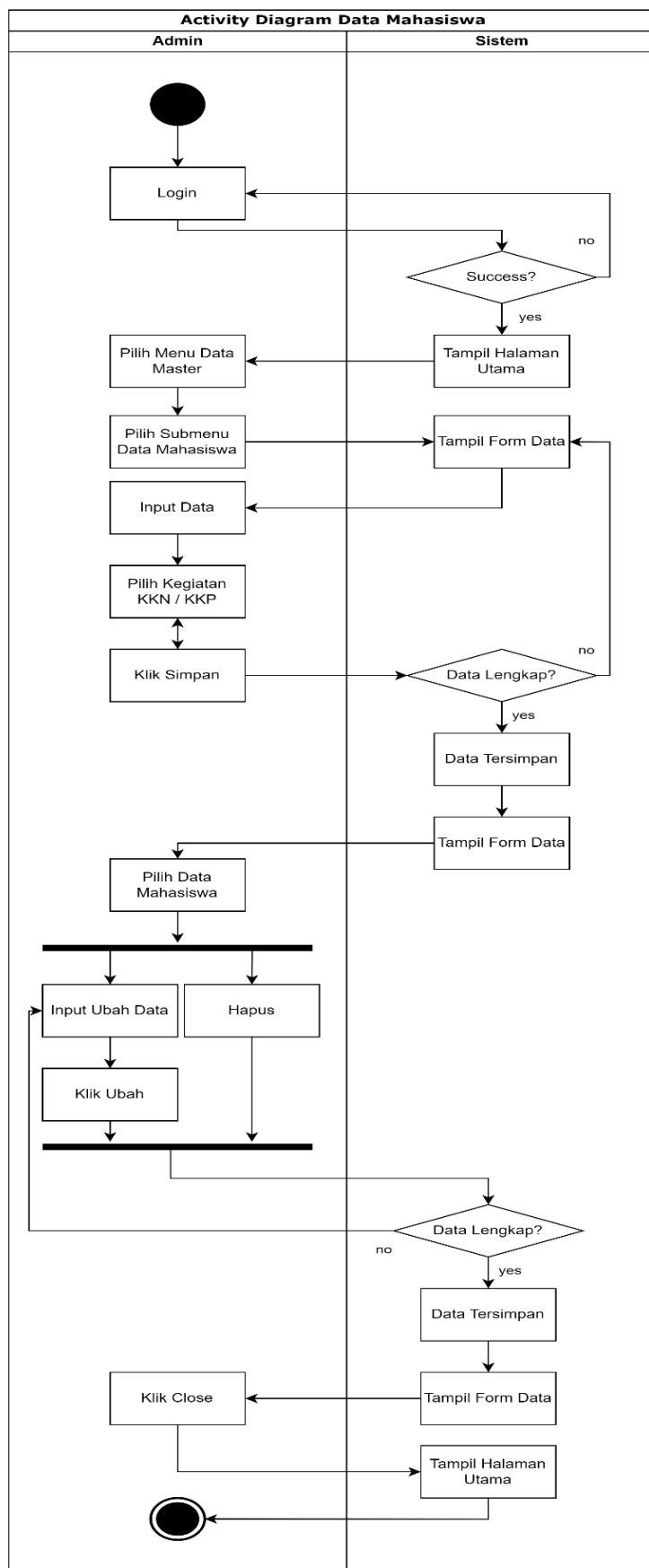
Pada bagian Activity Diagram ini para aktor memiliki aktivitas dalam sistemnya masing-masing. Seperti yang akan digambarkan, bagaimana aktivitas dalam sistem akan berjalan. Aktivitas dibagi menjadi 3 aktivitas (Admin, Mahasiswa, dan Dosen. Adapun aktivitas yg digambarkan adalah sebagai berikut :

#### 1. Admin

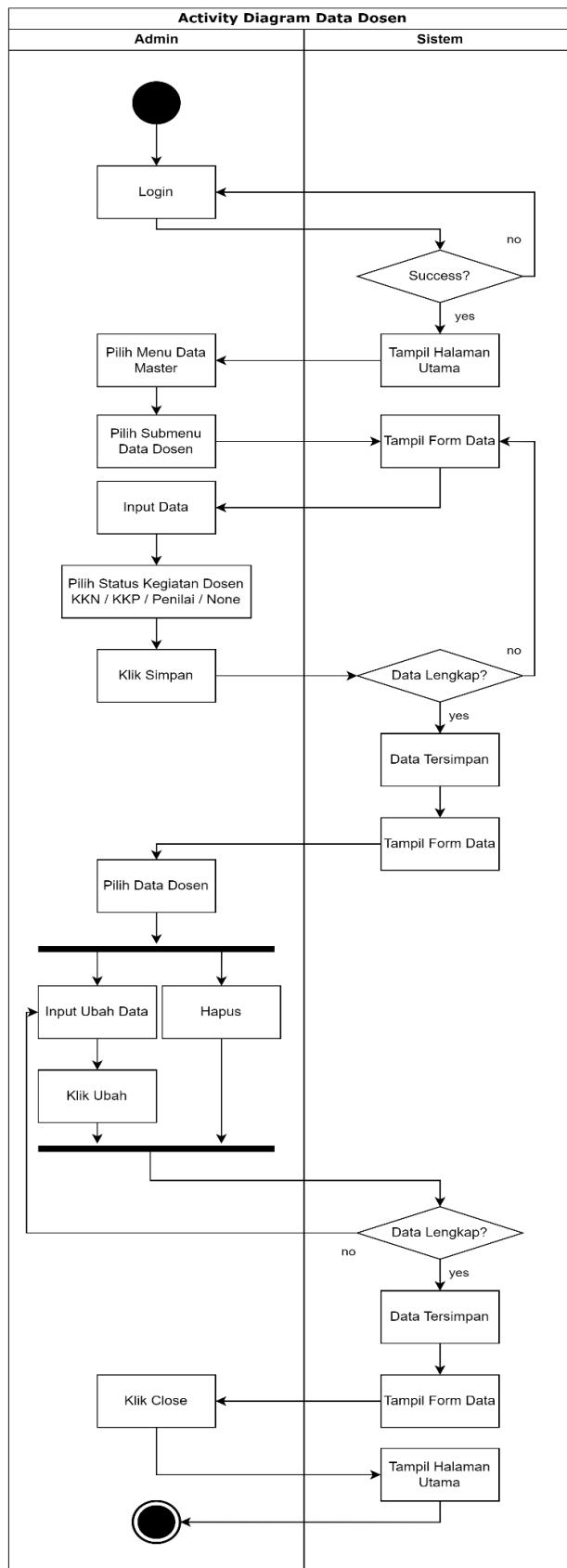
- Login



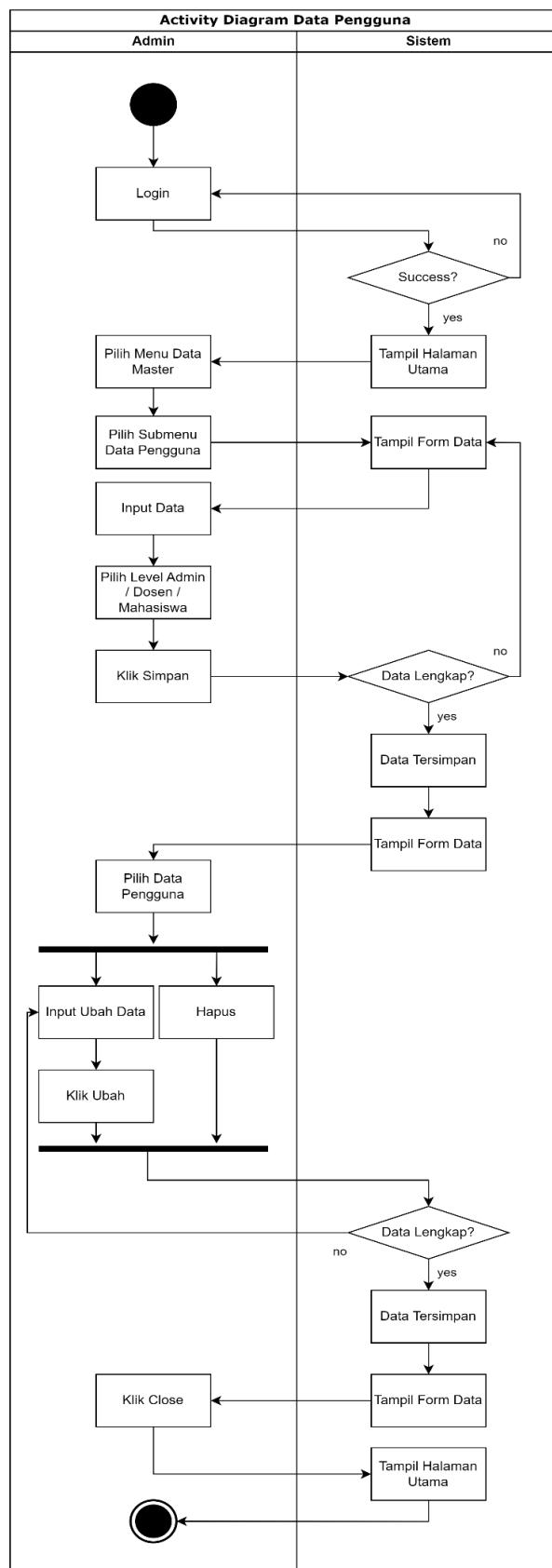
b. Data Master Mahasiswa



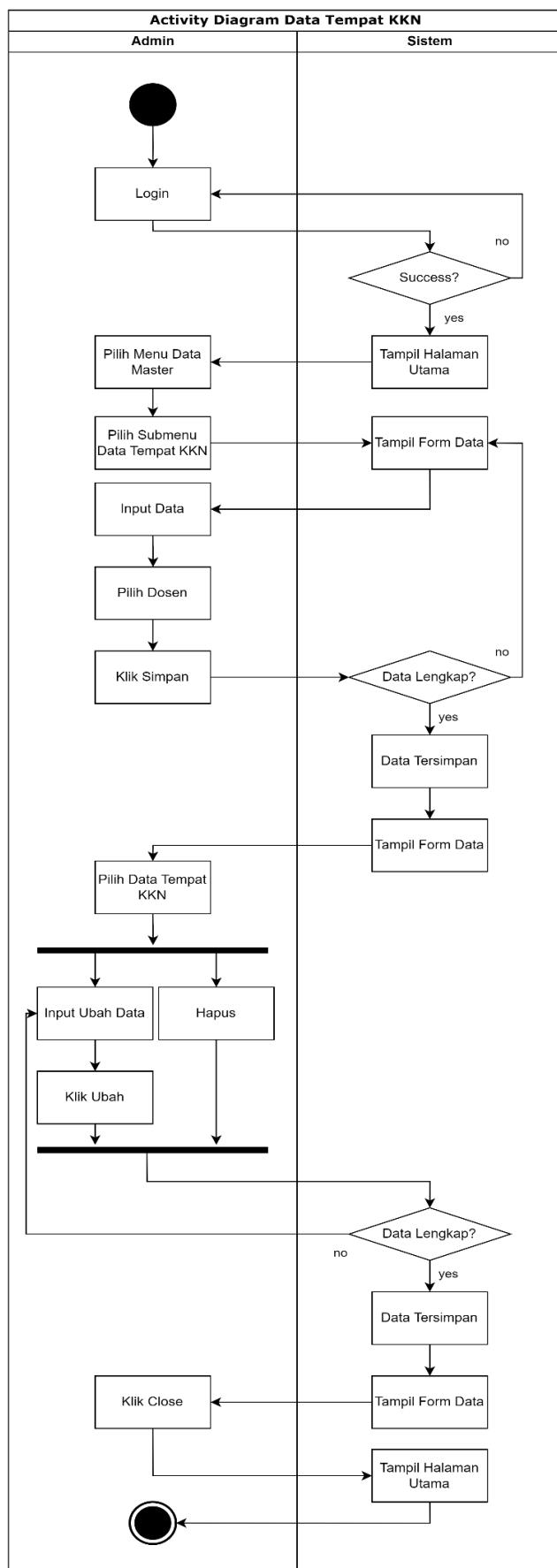
c. Data Master Dosen



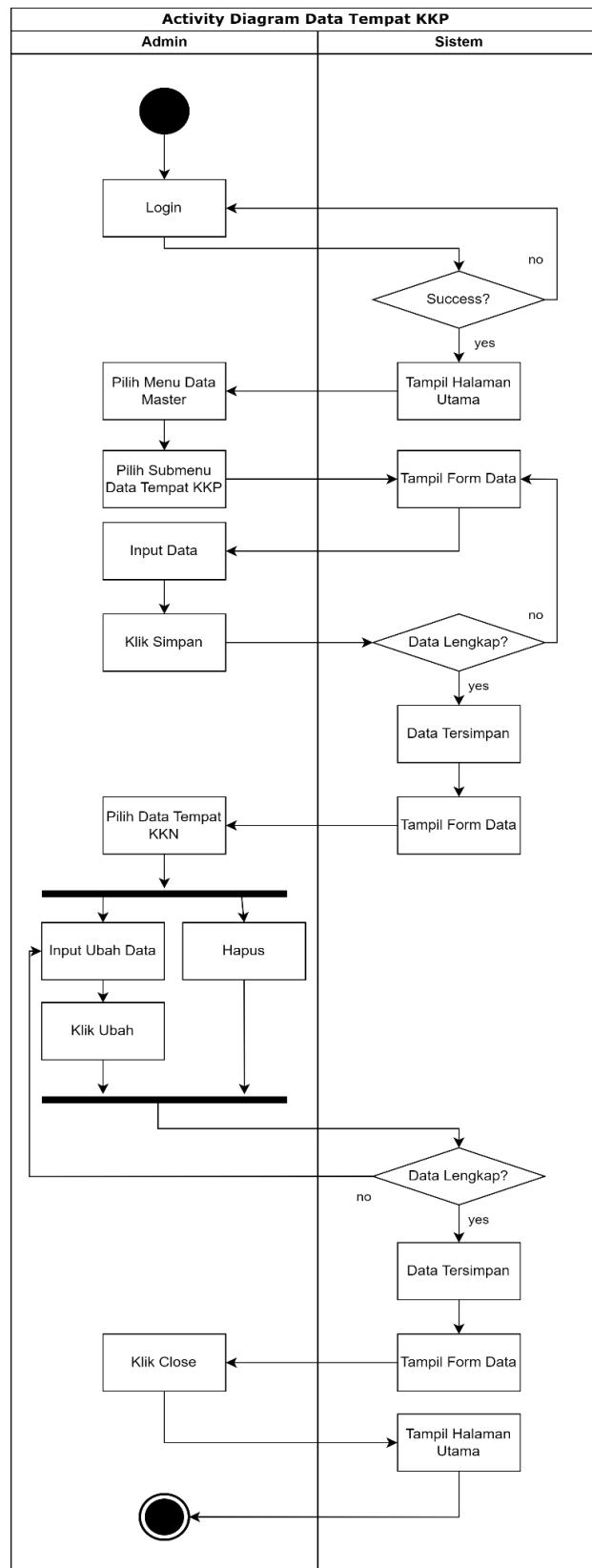
#### d. Data Master Pengguna



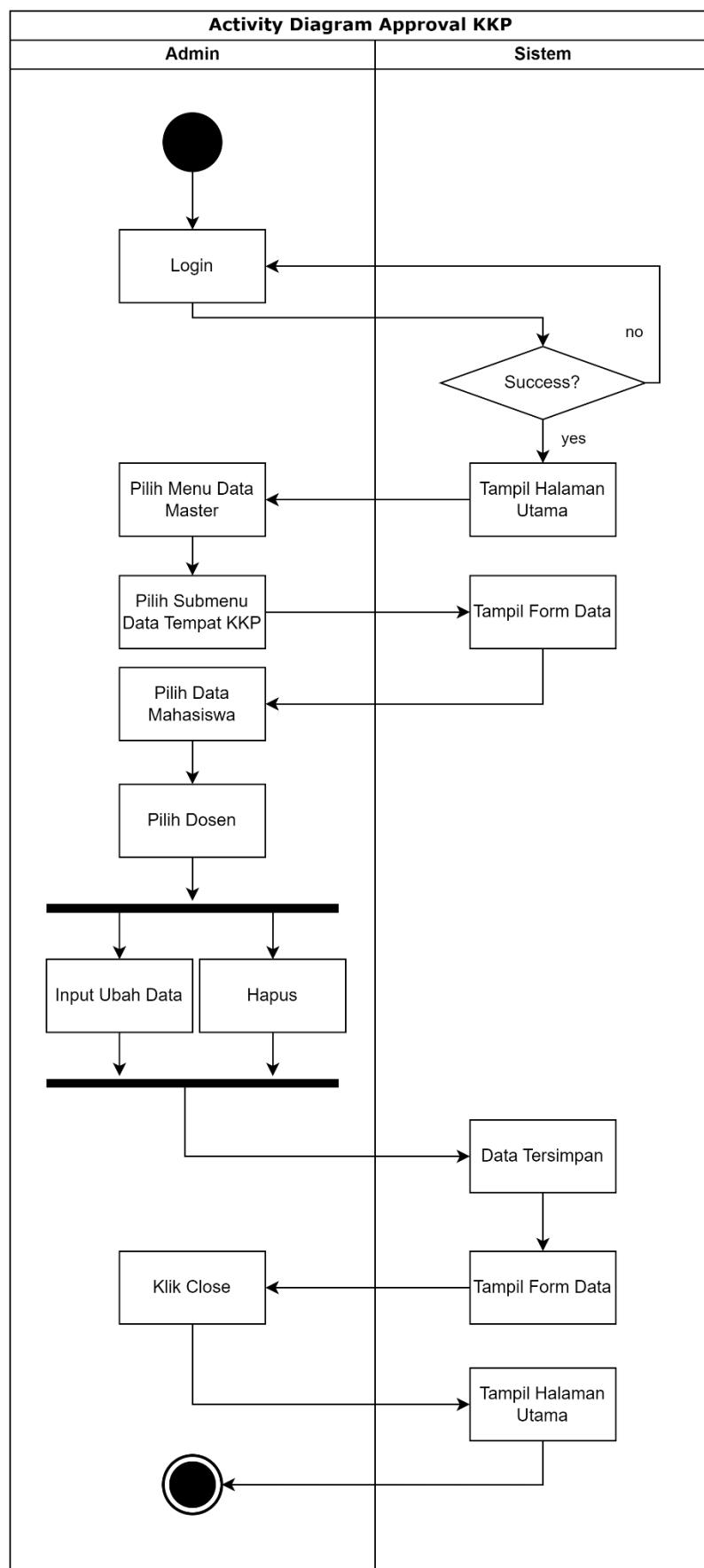
e. Data Master Tempat KKN



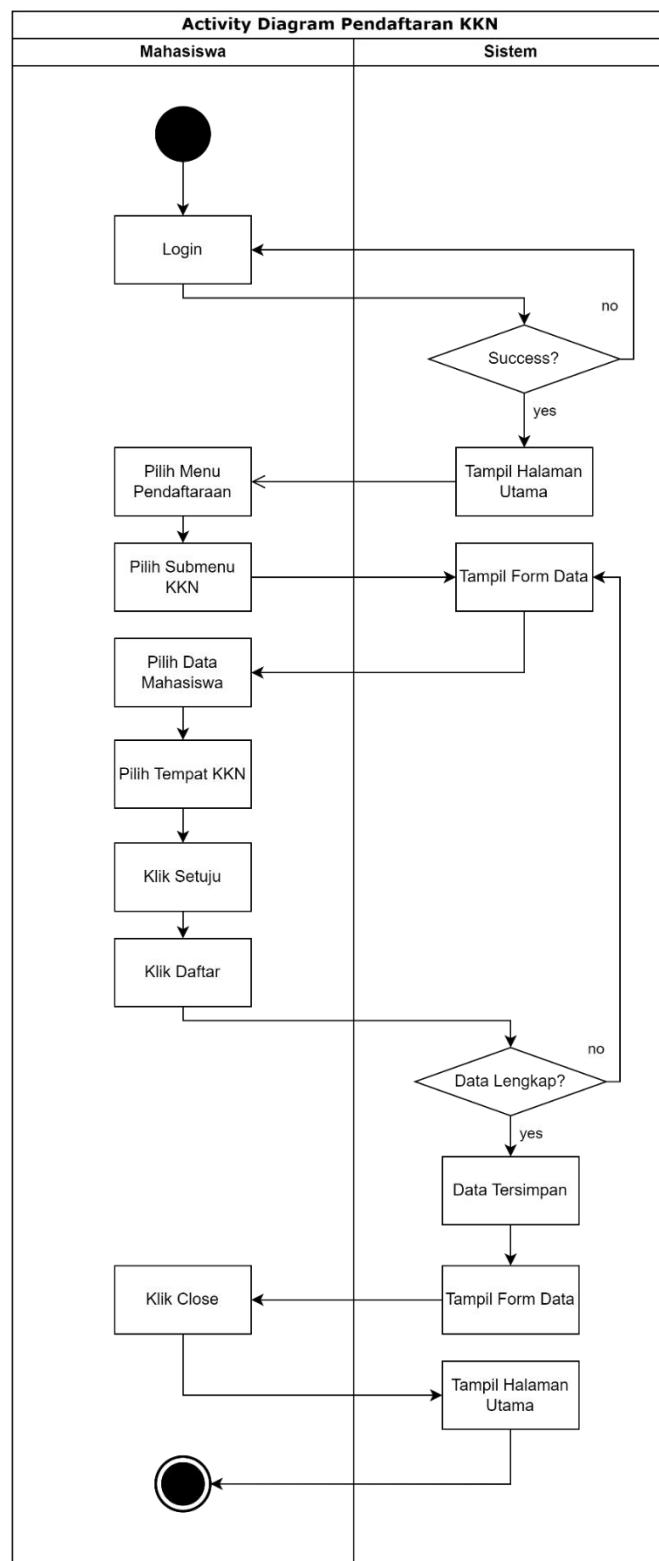
f. Data Master Tempat KKP



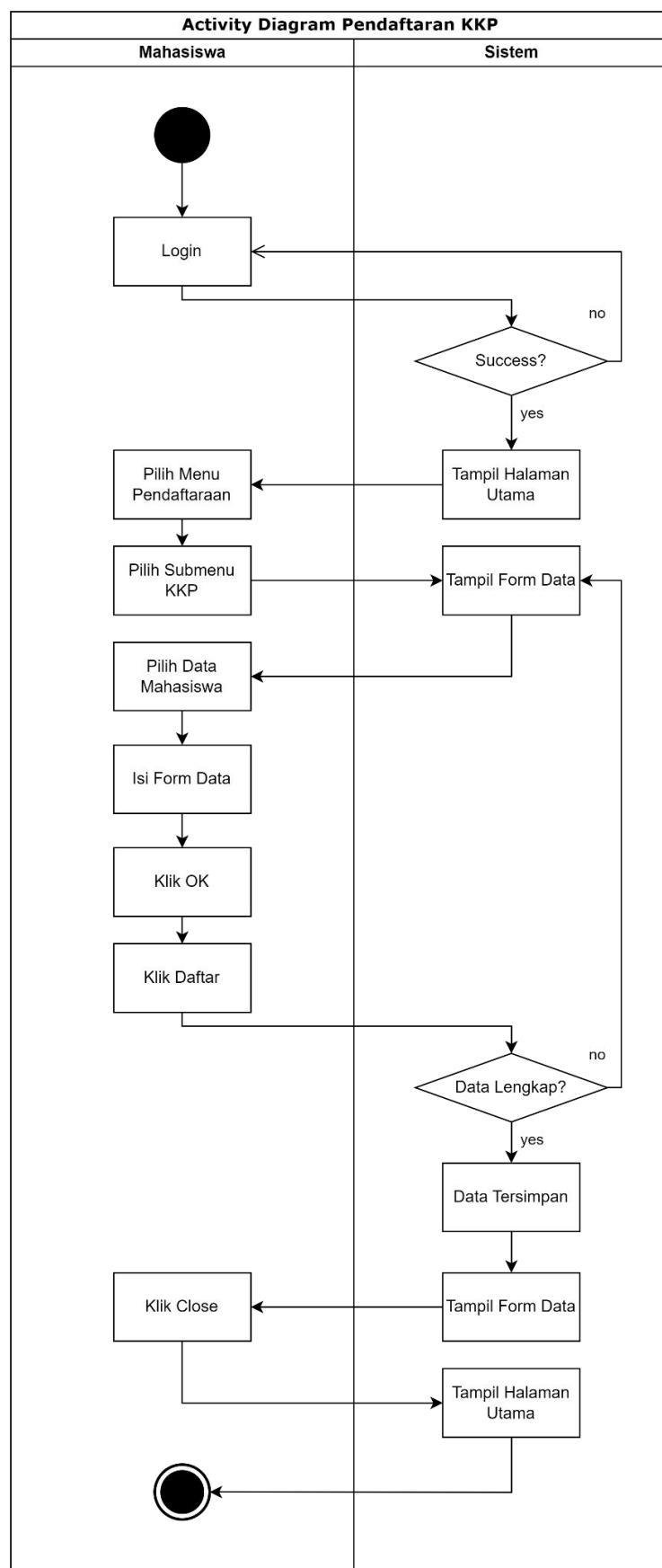
g. Approval Tempat KKP



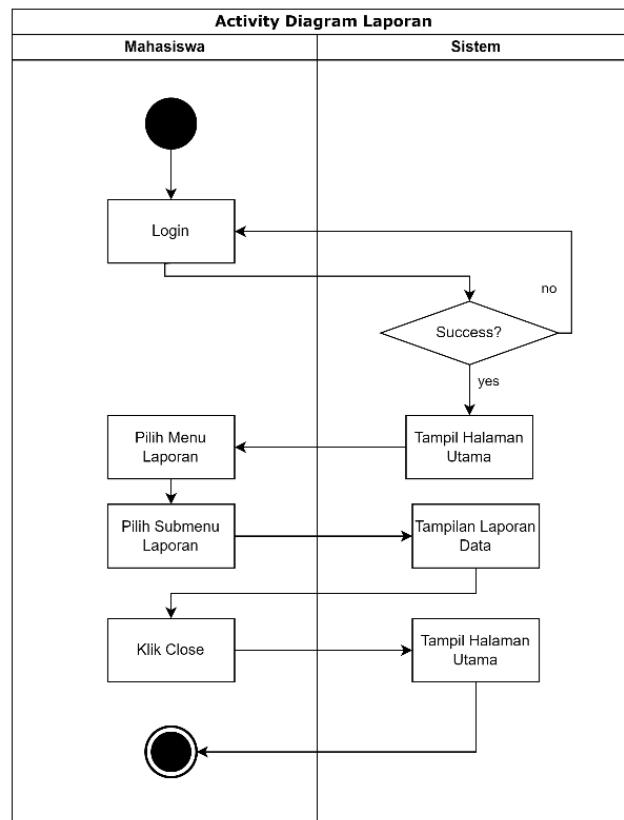
### h. Data Pendaftaran KKN



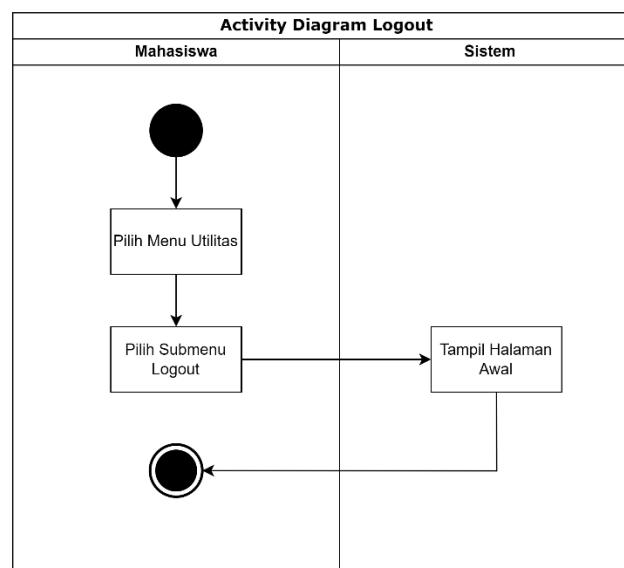
i. Data Pendaftaran KKP



j. Data Laporan

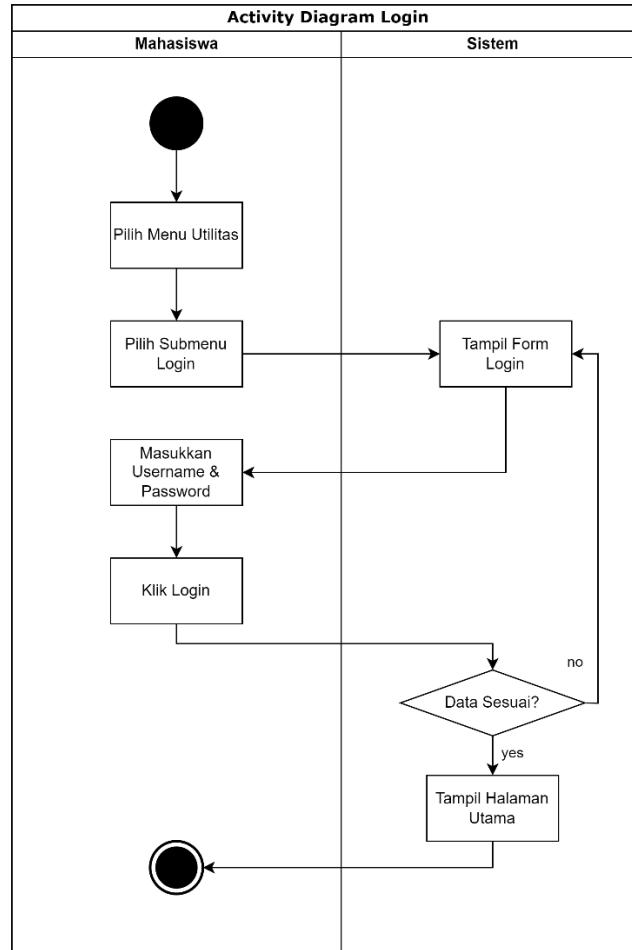


k. Logout

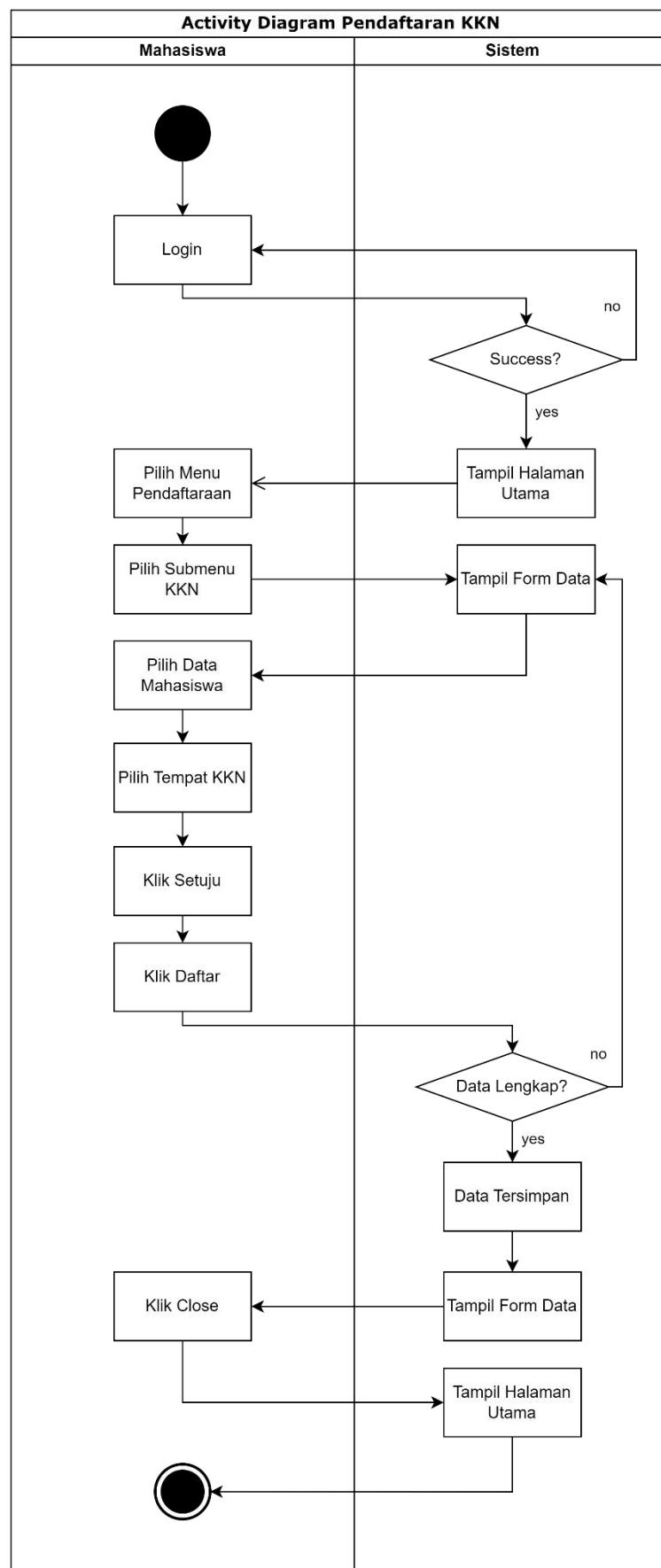


## 2. Mahasiswa

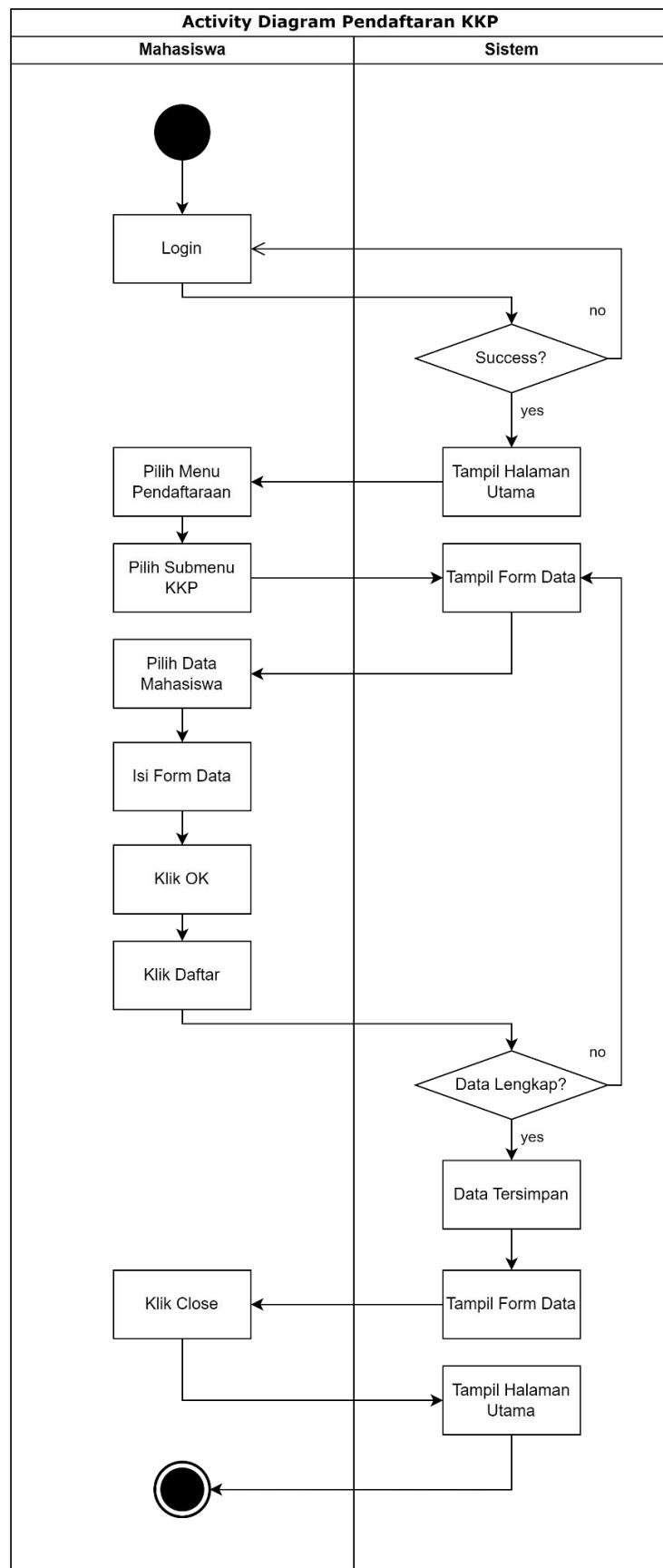
### a. Login



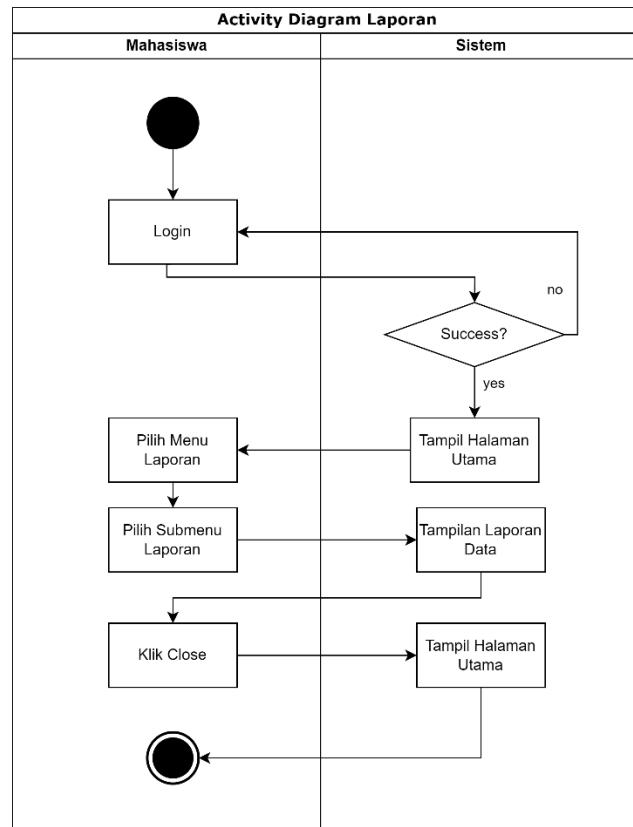
b. Pendaftaran KKN



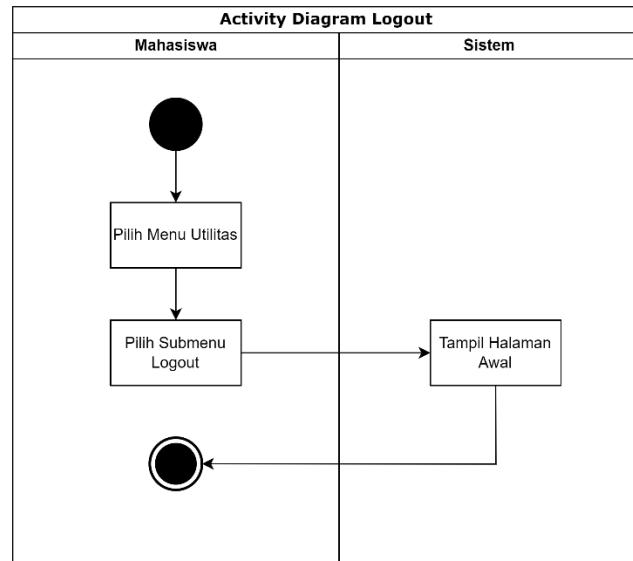
c. Pendaftaran KKP



d. Laporan Data Mahasiswa Terdaftar

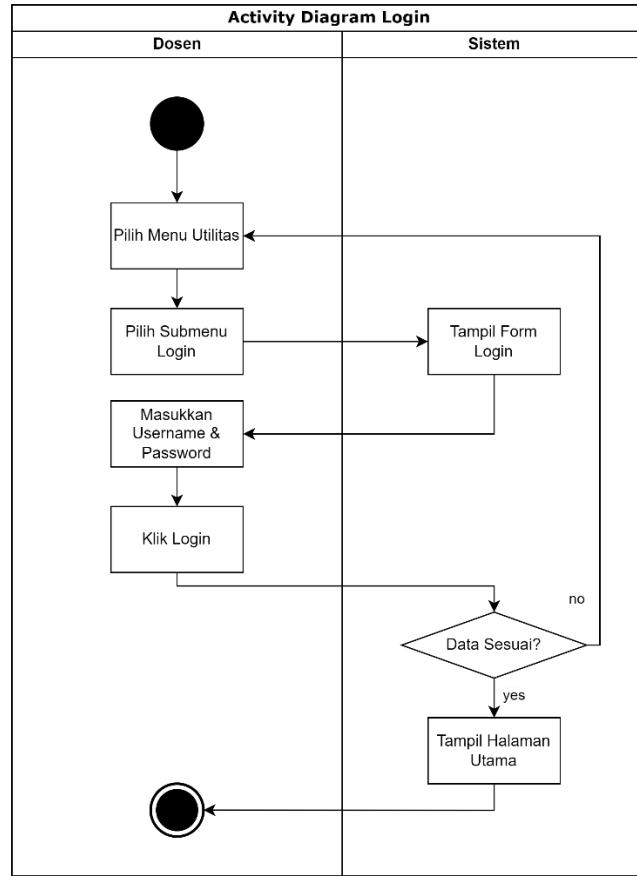


e. Logout

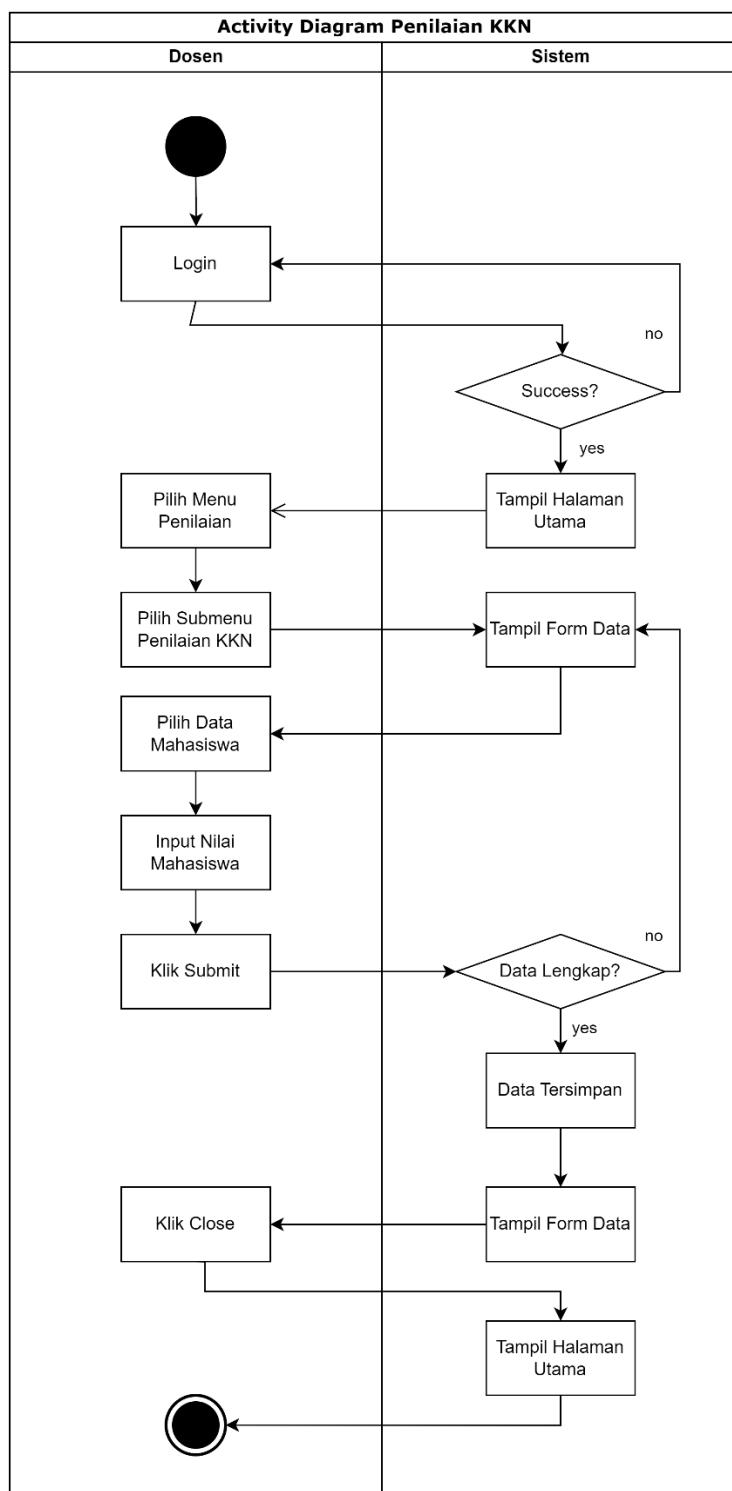


### 3. Dosen

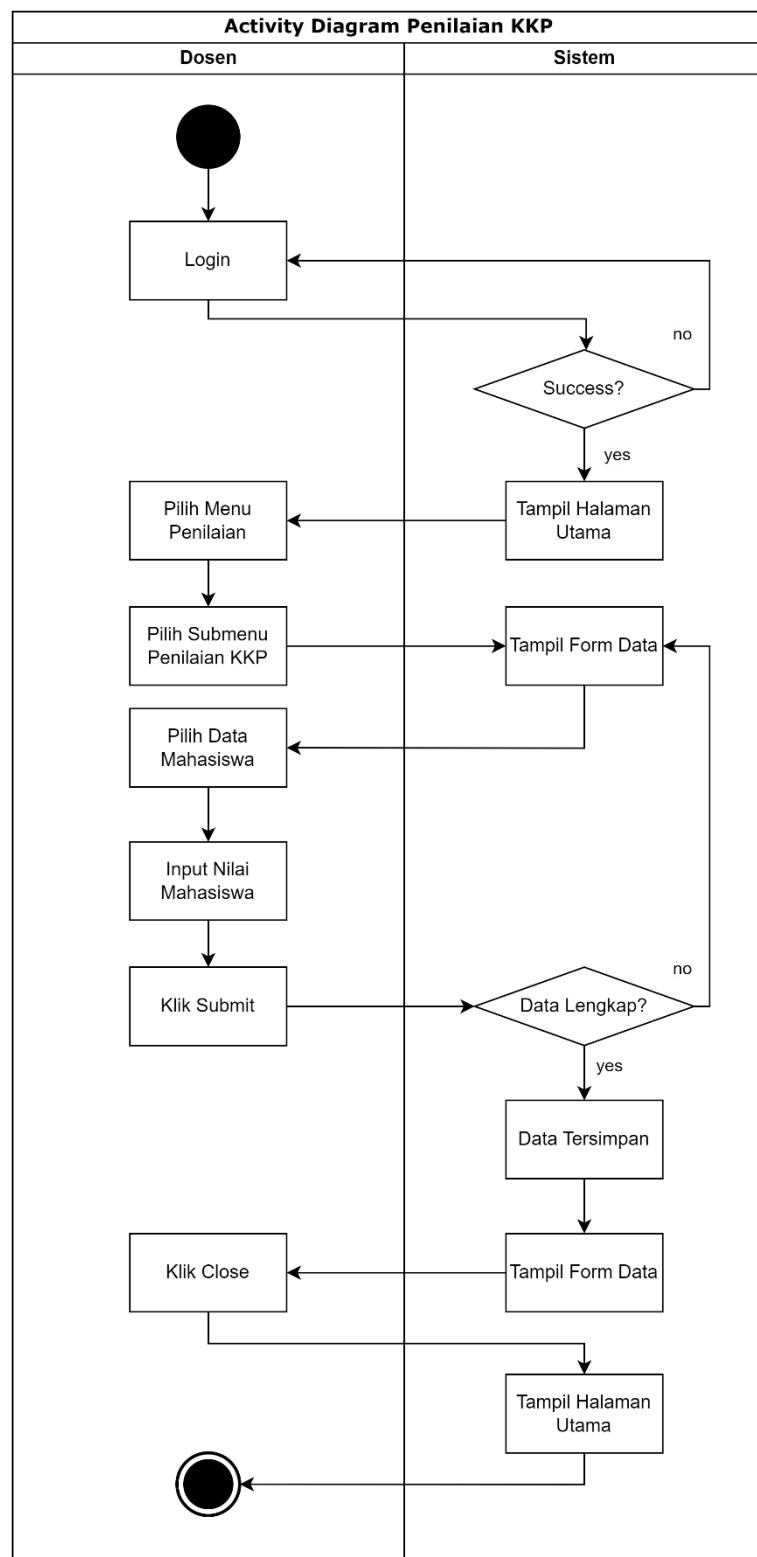
#### a. Login



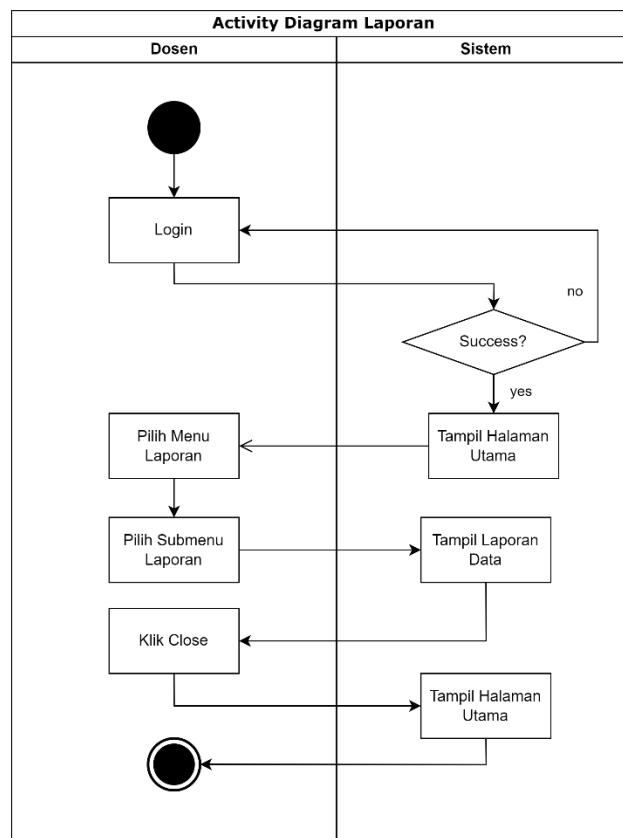
b. Penilaian KKN



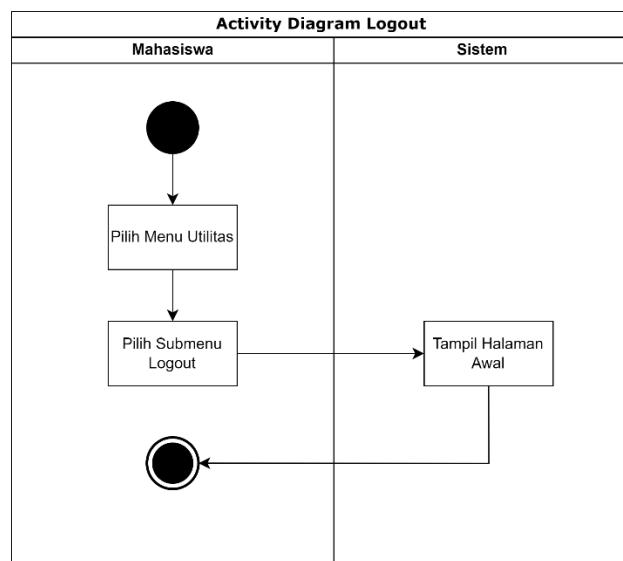
c. Penilaian KKP



d. Laporan Nilai Mahasiswa



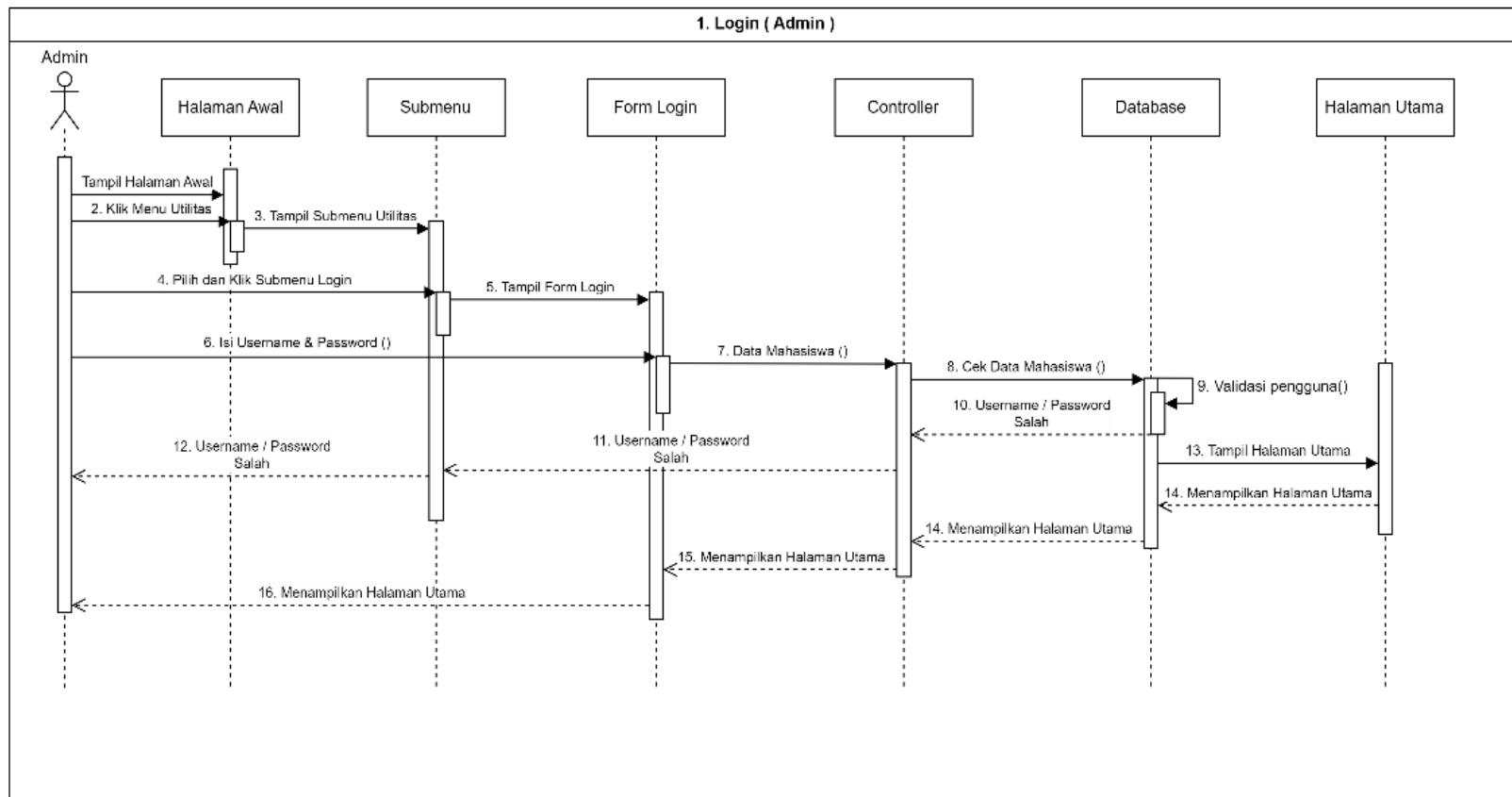
e. Logout



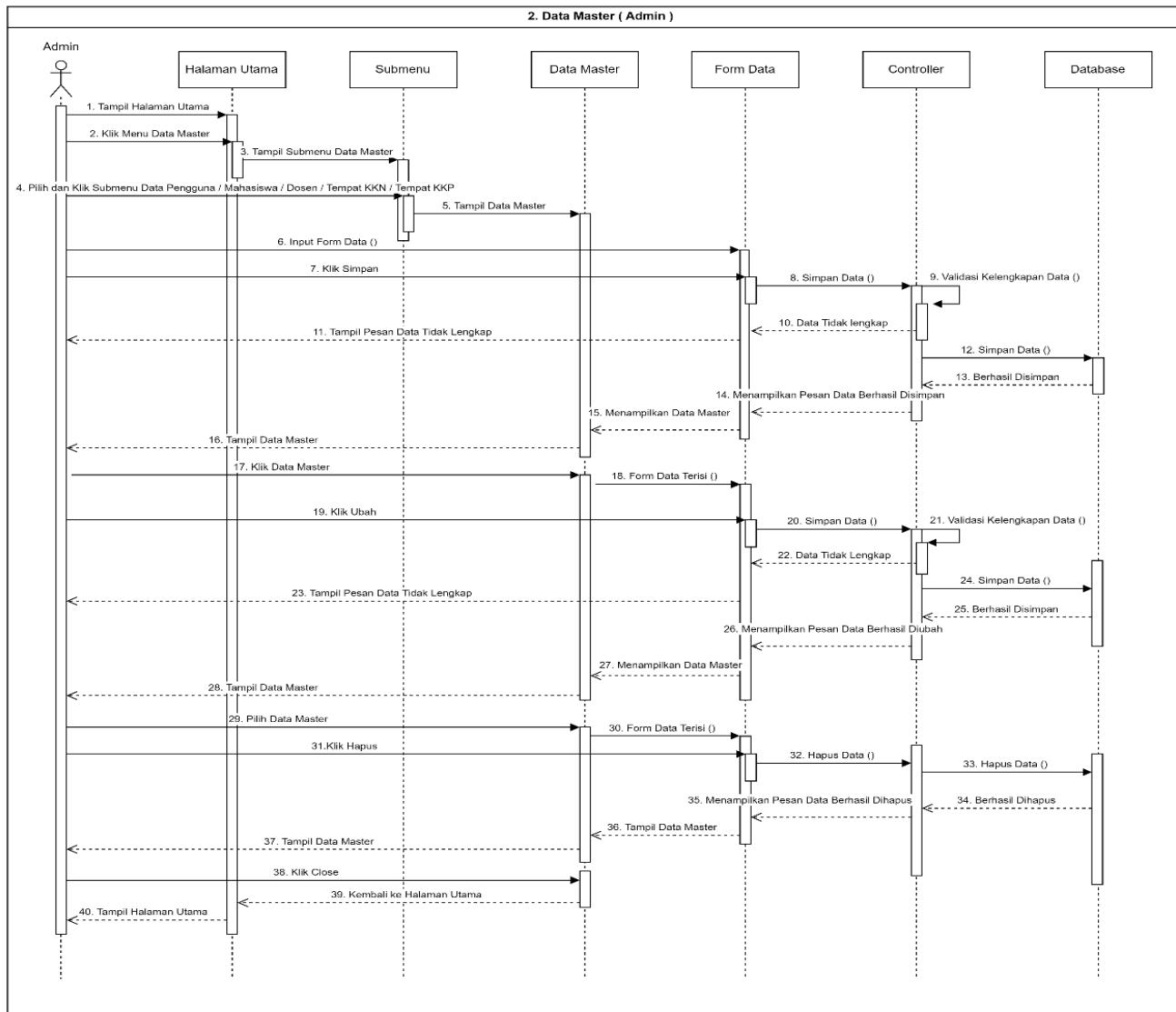
### 2.1.3 Sequence Diagram

#### 1. Admin

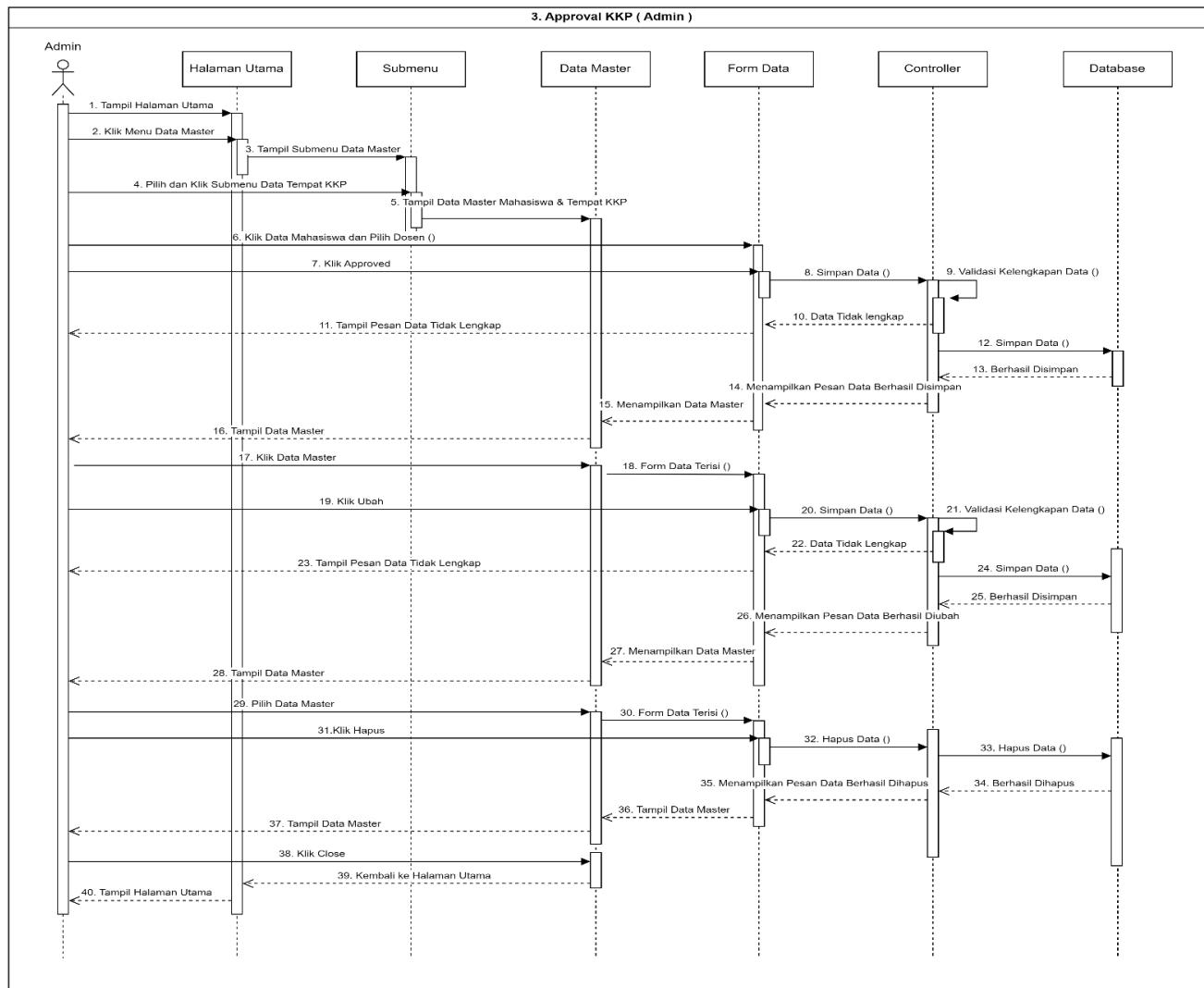
##### a. Login



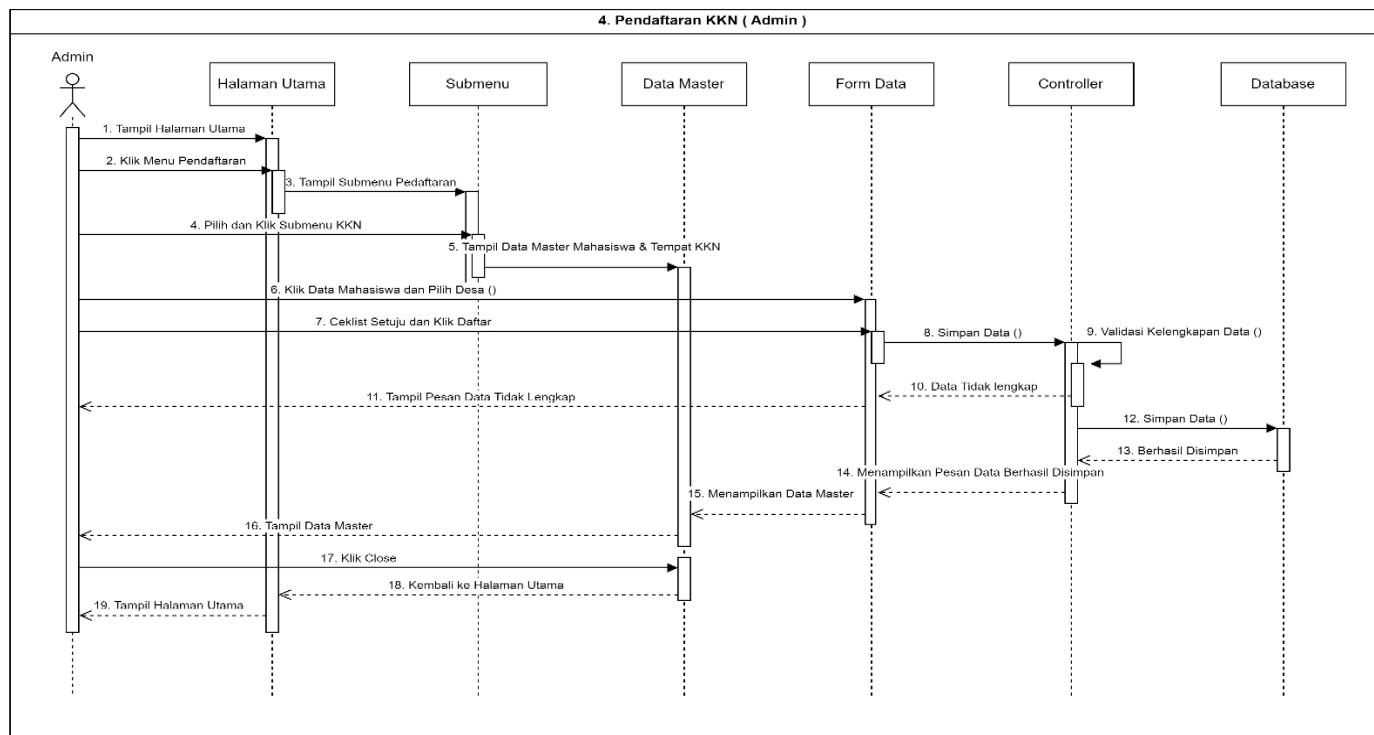
## b. Data Master



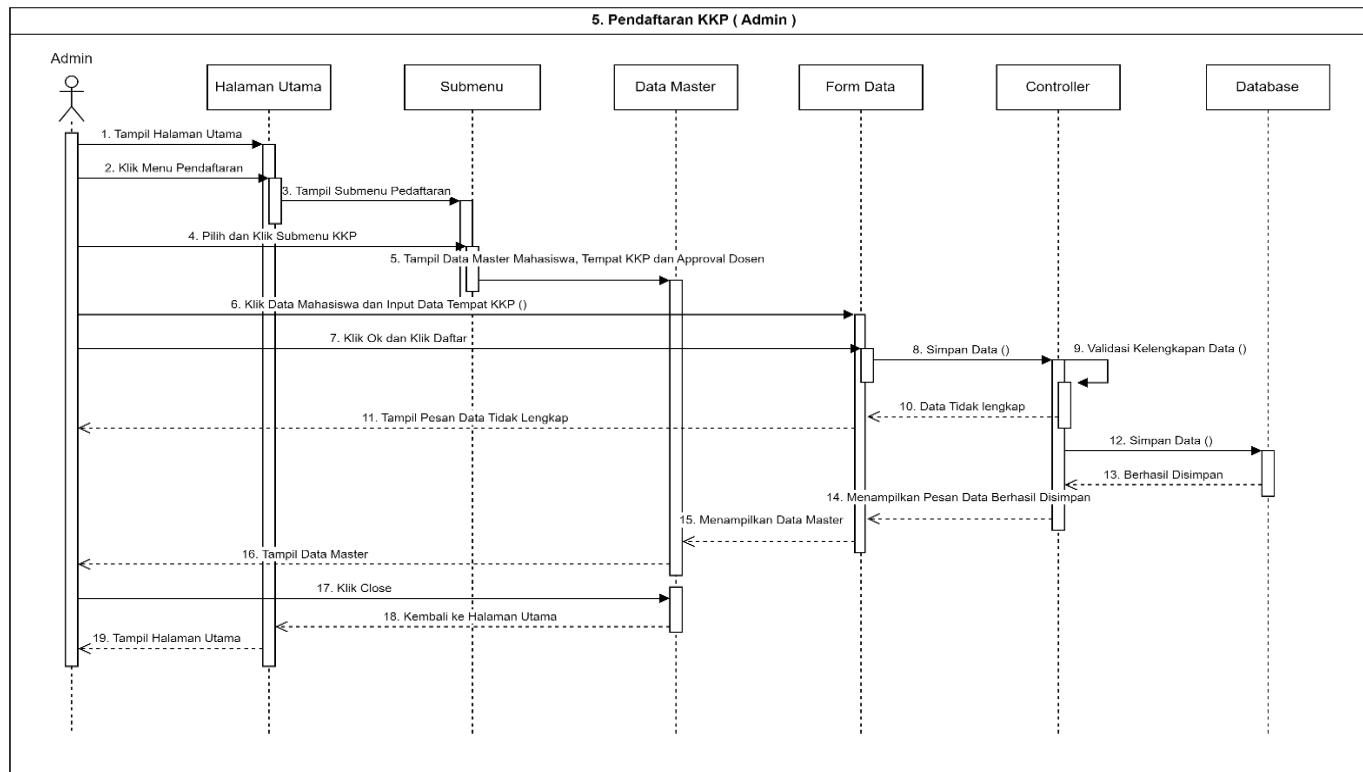
### c. Approval Tempat KKP



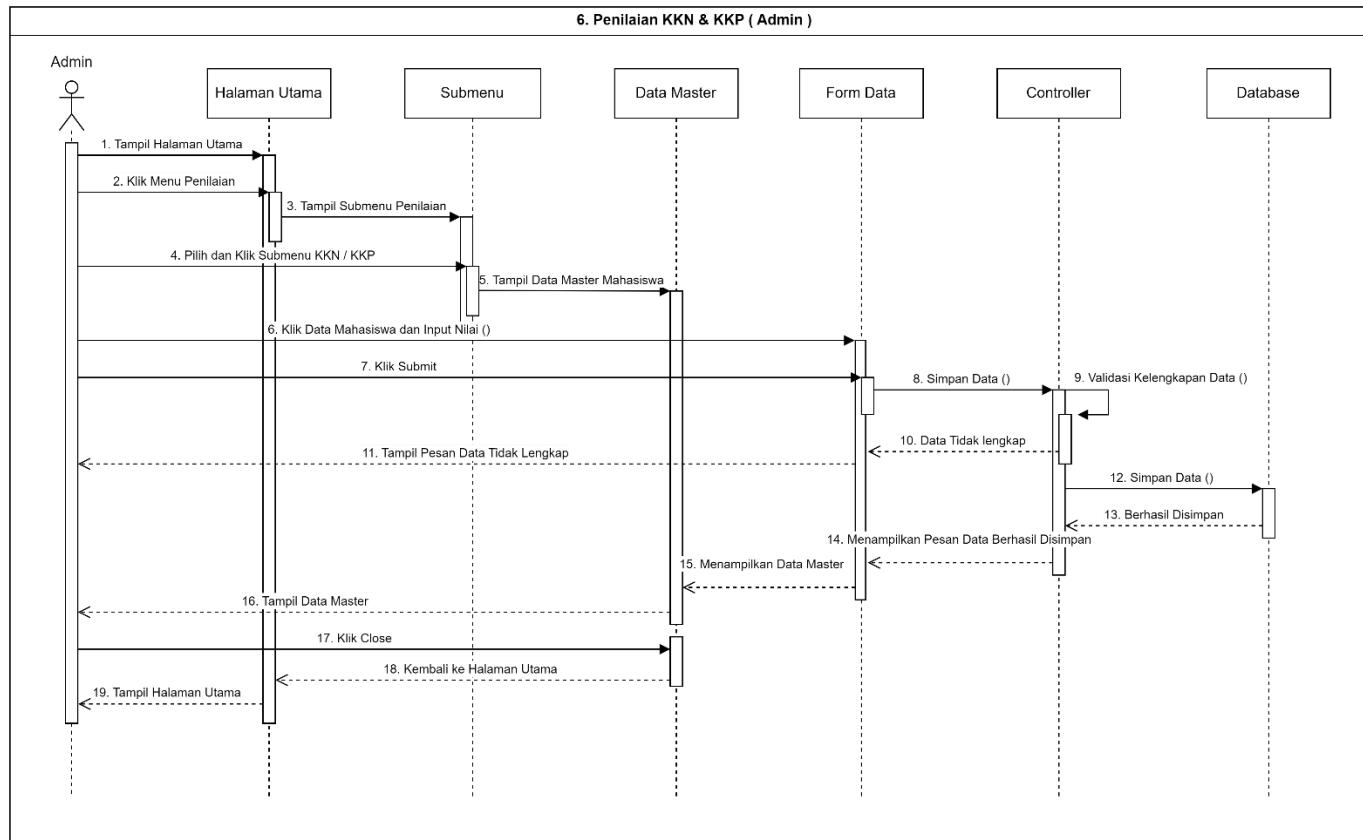
#### d. Pendaftaran KKN



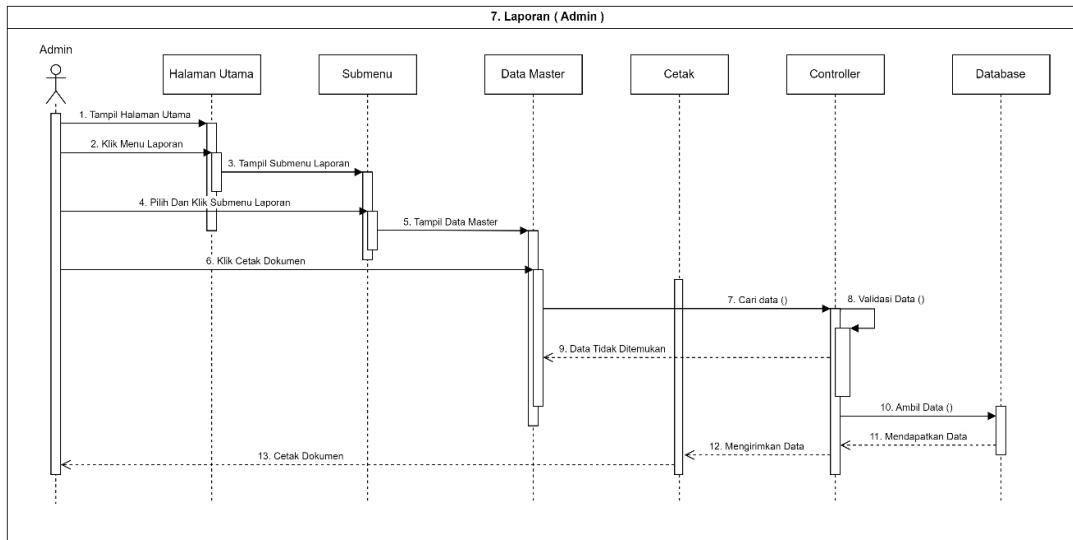
### e. Pendaftaran KKP



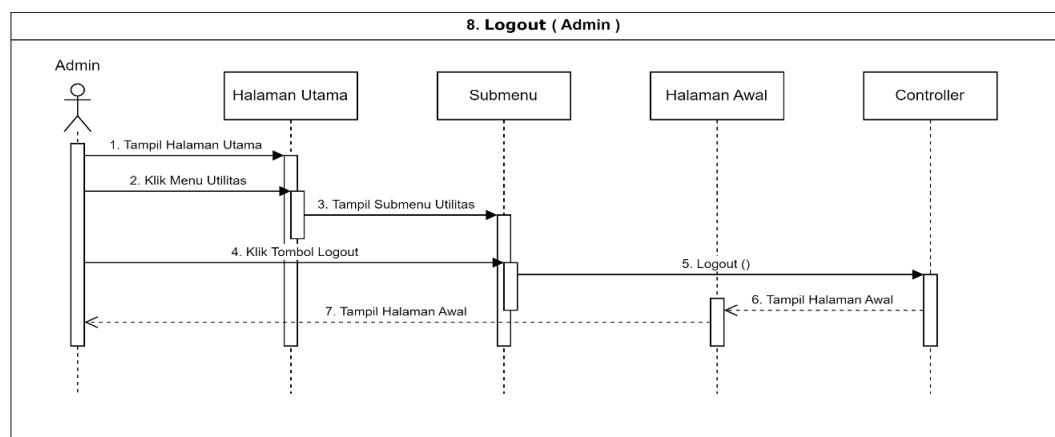
## f. Penilaian KKN – KKP



### g. Data Laporan

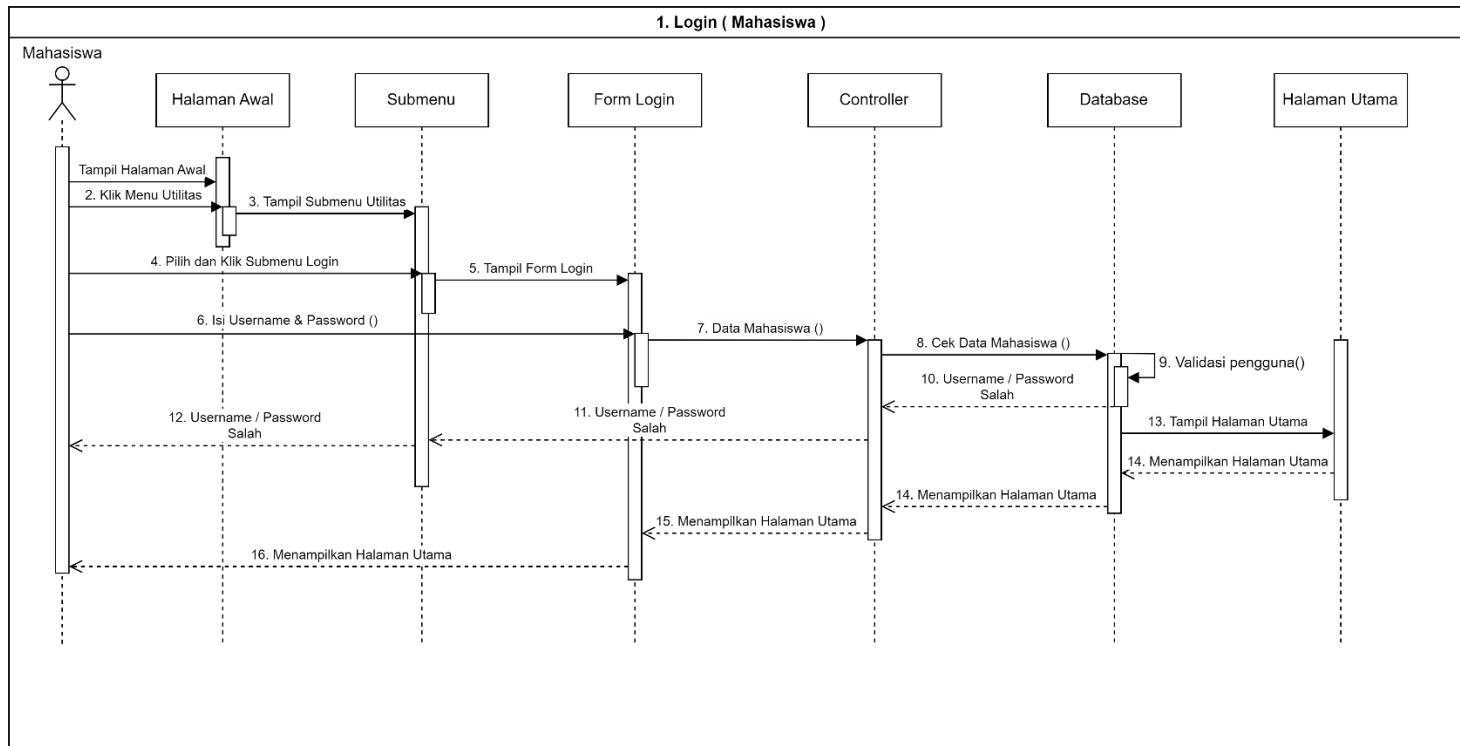


### h. Logout

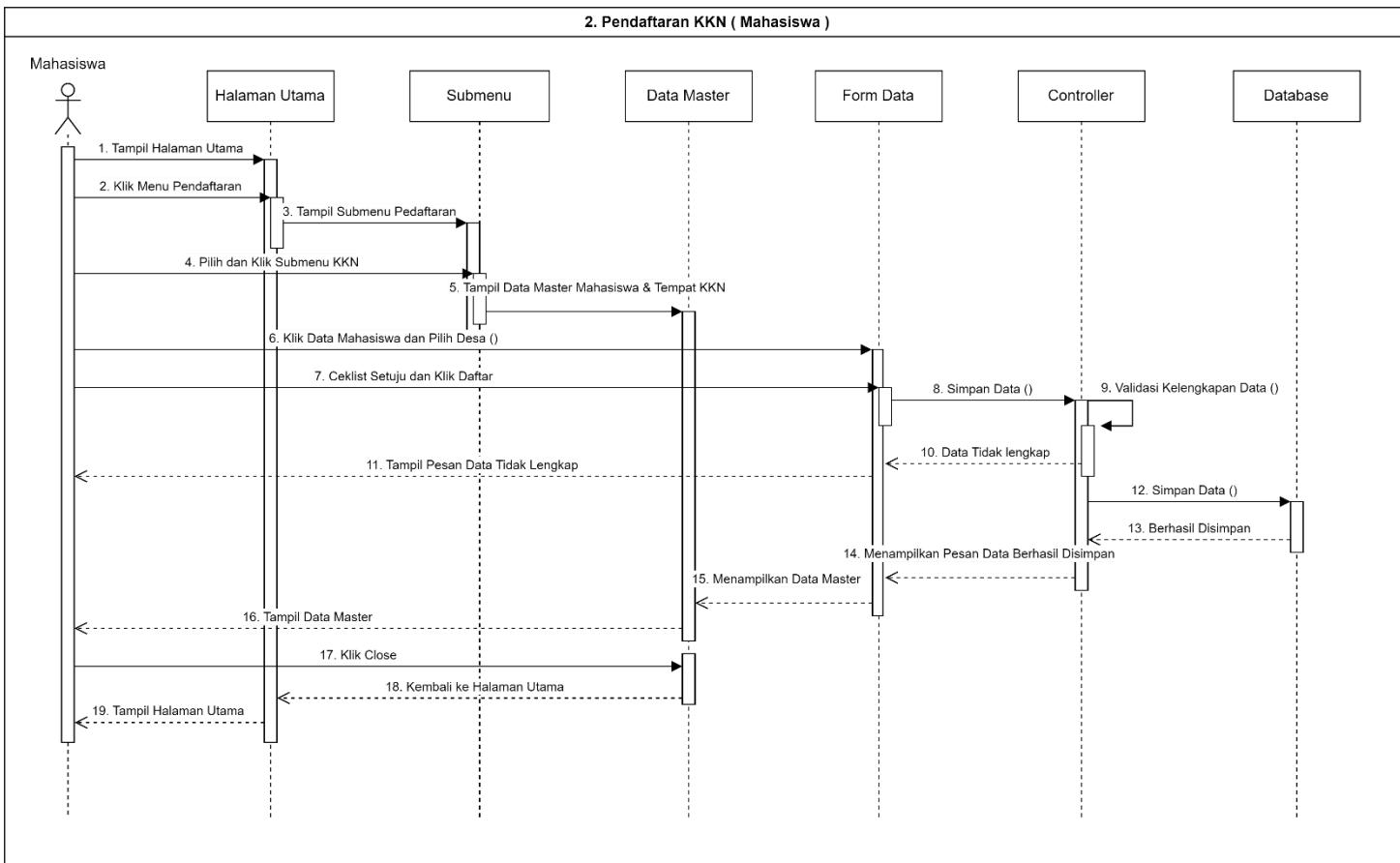


## 2. Mahasiswa

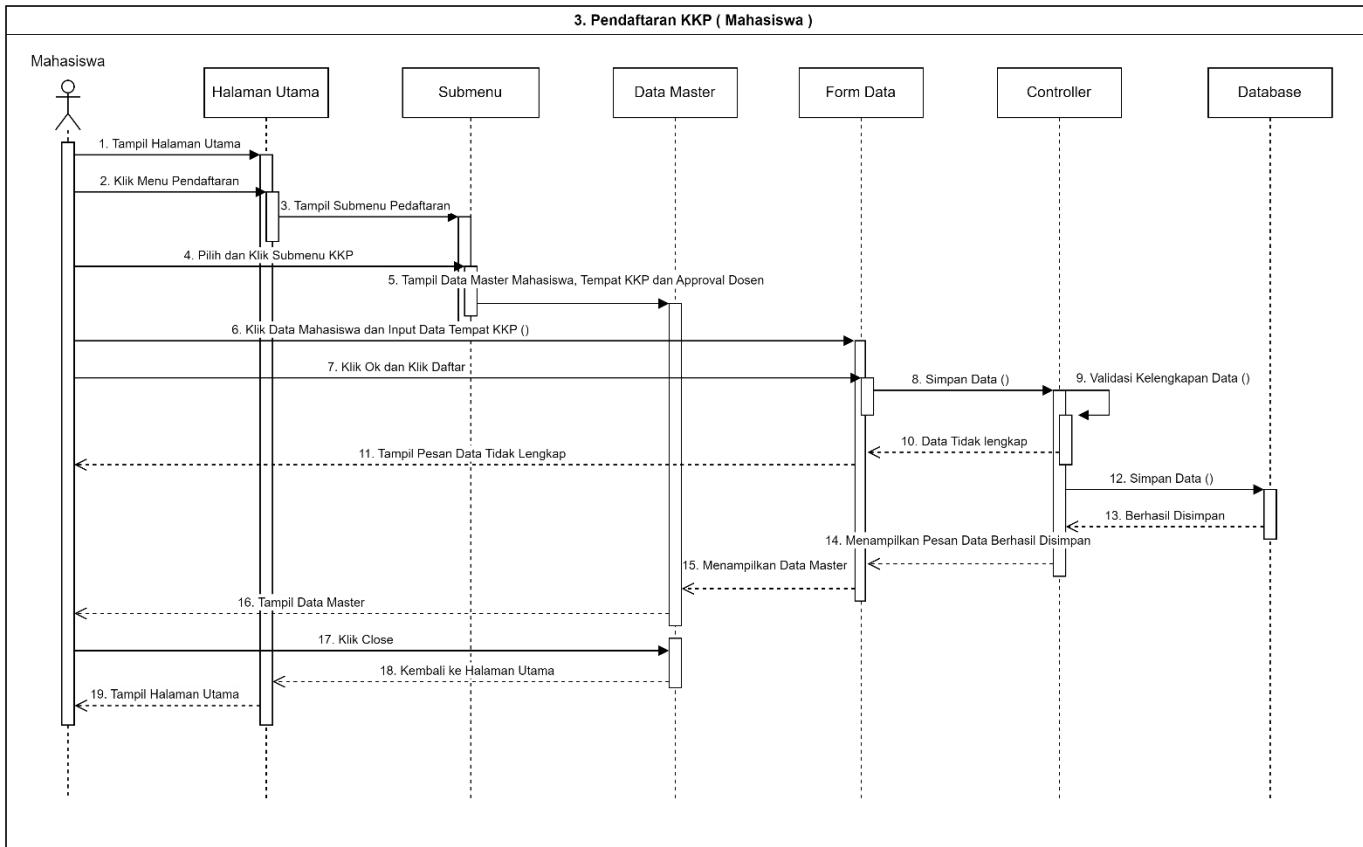
### a. Login



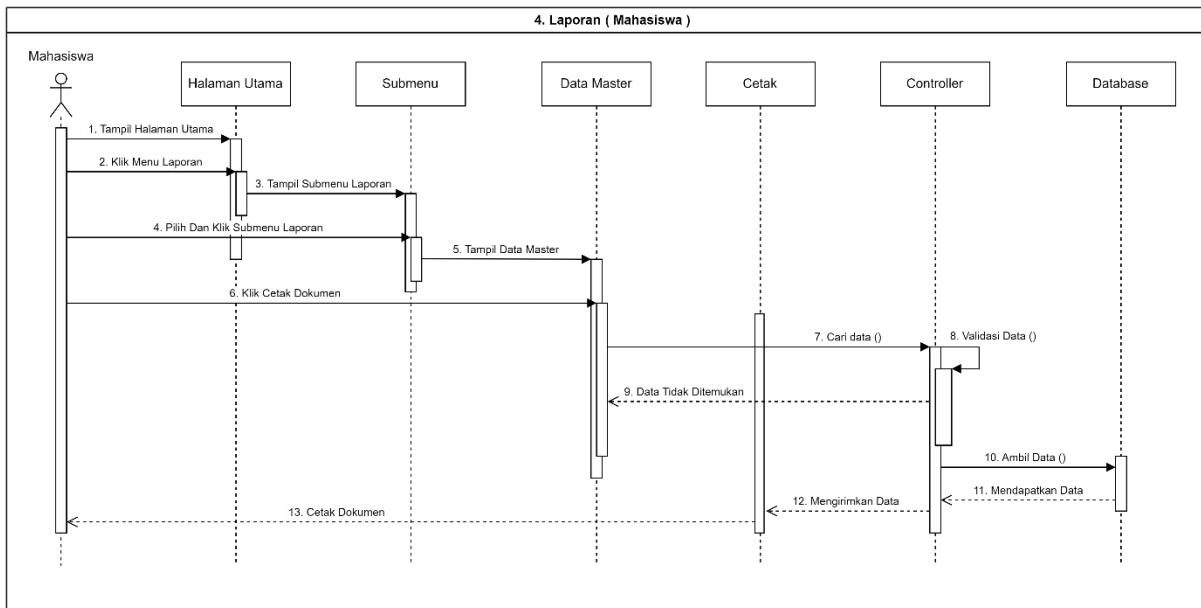
b. Pendaftaran KKN



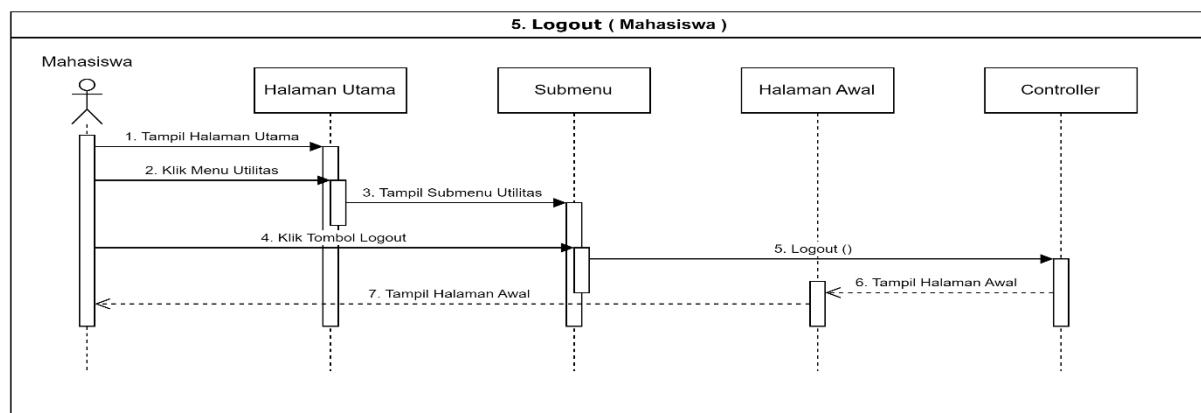
### c. Pendaftaran KKP



d. Laporan Mahasiswa Terdaftar

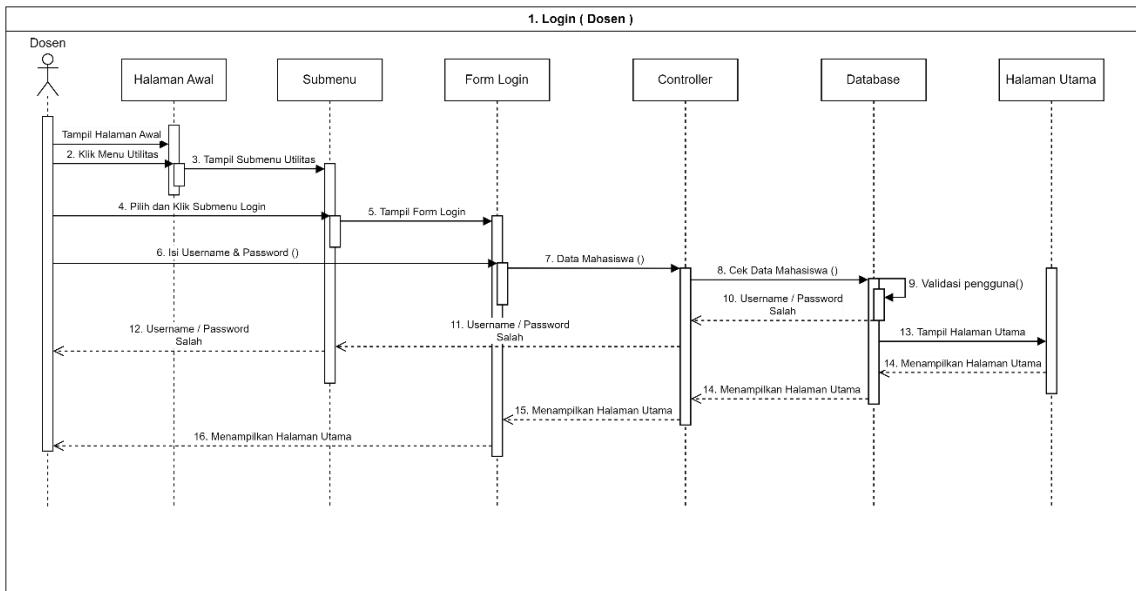


e. Logout

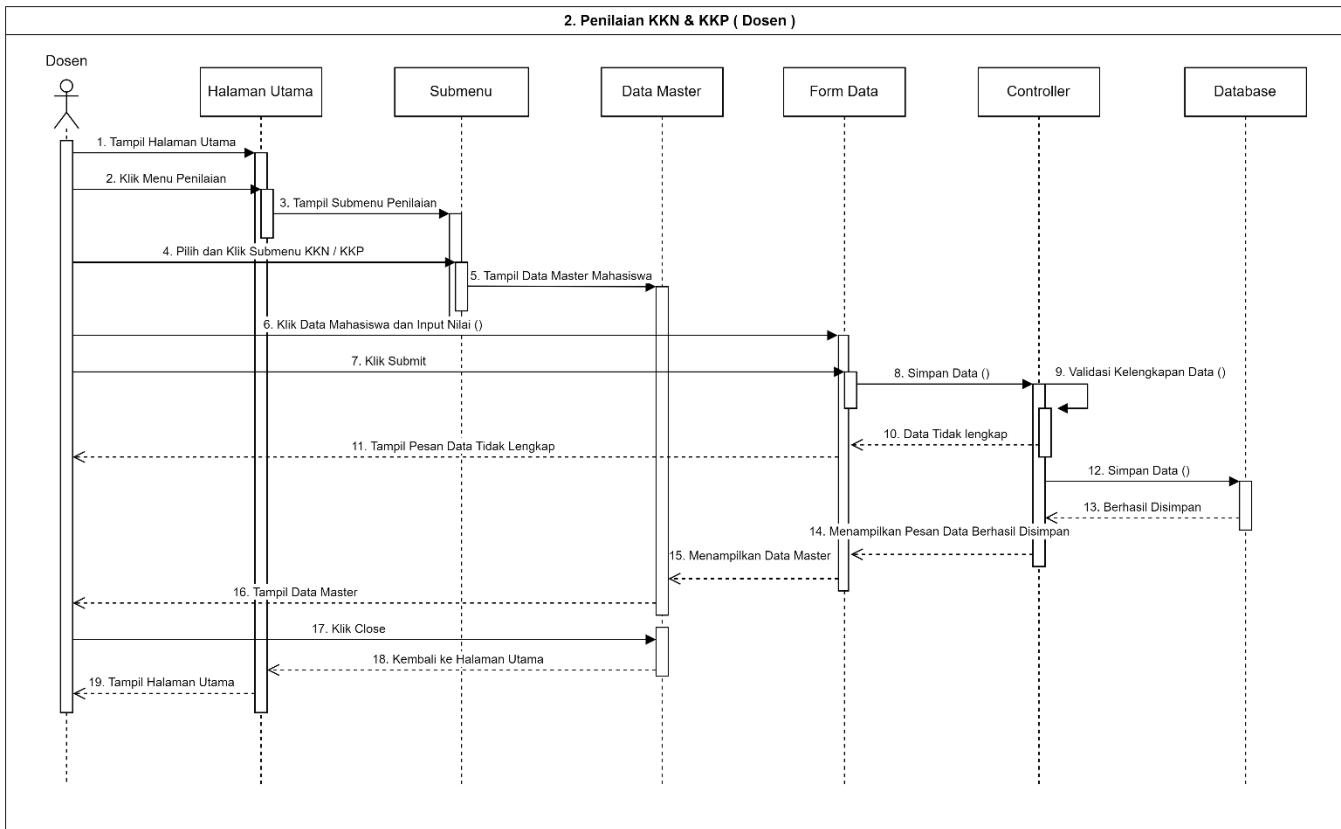


### 3. Dosen

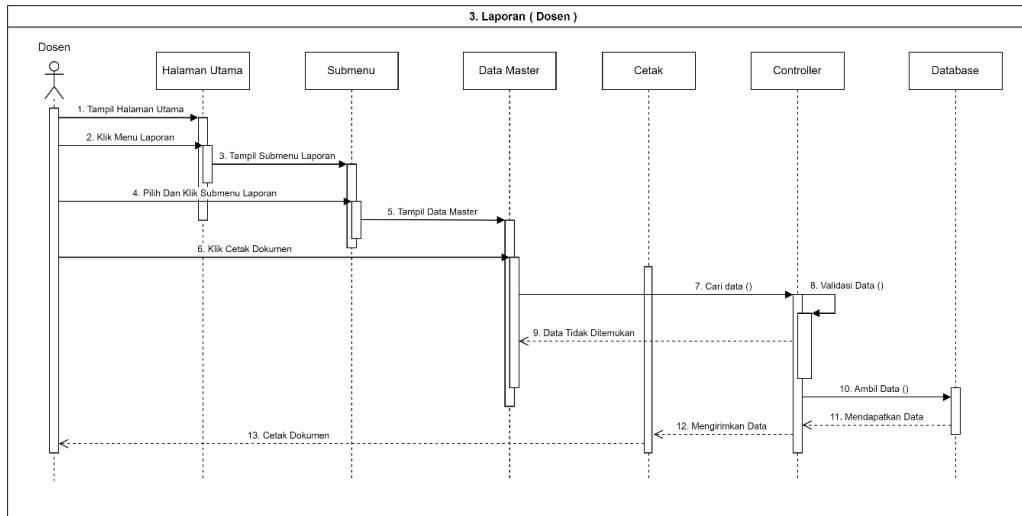
#### a. Login



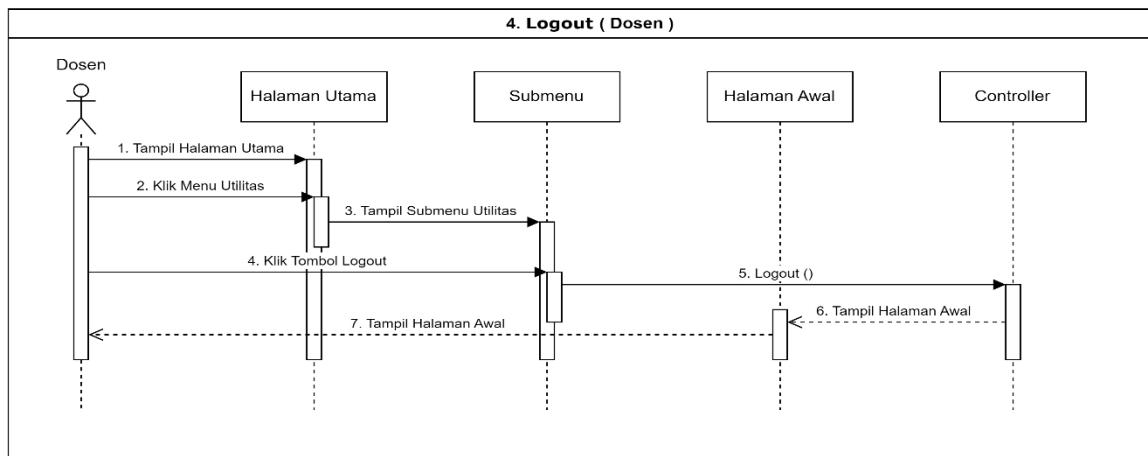
b. Penilaian KKN dan KKP



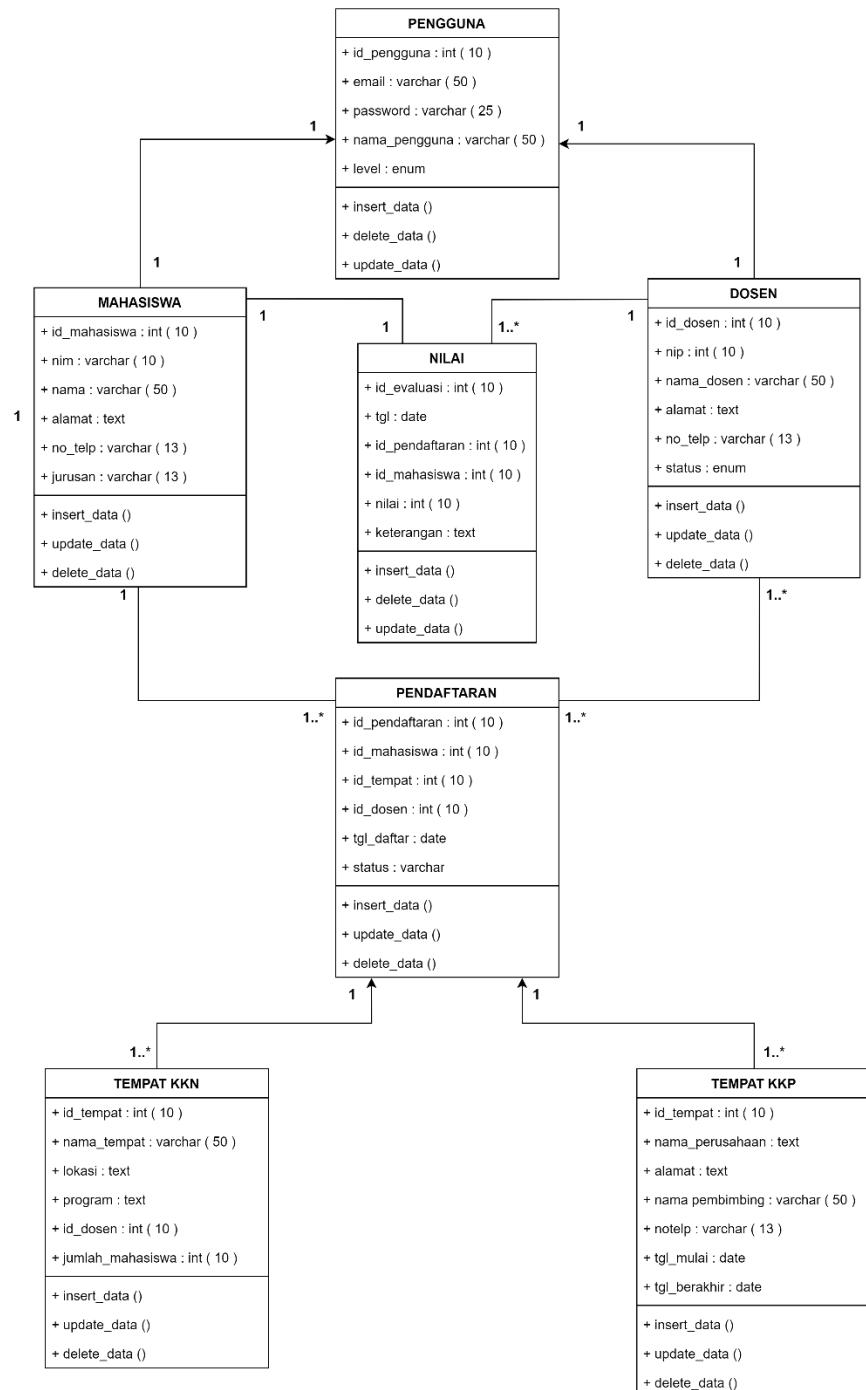
### c. Laporan



### d. Logout



## 2.1.4 Class Diagram



## 2.1.5. Membuat Database

### 1. Membuat Database pada Neatbeans

Sebelum membuat aplikasi pendaftaran, maka dari itu sebelumnya kita membuat dan merancang database, buat database dengan nama **db\_kkn**. kemudian buatlah table-table seperti berikut:

- a. Membuat data tabel “**dosen**”

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id_dosen</b>	int		No	None		AUTO_INCREMENT		
2	<b>nip</b>	int		No	None				
3	<b>nama_dosen</b>	varchar(50)	utf8mb4_0900_ai_ci	No	None				
4	<b>Alamat</b>	text	utf8mb4_0900_ai_ci	No	None				
5	<b>no_telp</b>	varchar(13)	utf8mb4_0900_ai_ci	No	None				
6	<b>status</b>	enum('none', 'kkn', 'kkp', 'pengujji/penilai')	utf8mb4_0900_ai_ci	Yes	NULL				

- b. Membuat data tabel evaluasi kkn “**evaluasi**”

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id_evaluasi</b>	int		No	None		AUTO_INCREMENT		
2	<b>tgl</b>	date		No	None				
3	<b>id_pendaftaran</b>	int		No	None				
4	<b>id_mahasiswa</b>	int		No	None				
5	<b>nilai</b>	int		No	None				
6	<b>keterangan</b>	text	utf8mb4_0900_ai_ci	No	None				

- c. Membuat data tabel “**evaluasi\_kkp**”

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id_evaluasi</b>	int		No	None		AUTO_INCREMENT		
2	<b>tgl</b>	date		No	None				
3	<b>id_pendaftaran</b>	int		No	None				
4	<b>id_mahasiswa</b>	int		No	None				
5	<b>nilai</b>	int		No	None				
6	<b>keterangan</b>	text	utf8mb4_0900_ai_ci	No	None				

- d. Membuat data table “**mahasiswa**”

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id_mahasiswa</b>	int		No	None		AUTO_INCREMENT		
2	<b>nim</b>	varchar(10)	utf8mb4_0900_ai_ci	No	None				
3	<b>nama</b>	varchar(50)	utf8mb4_0900_ai_ci	No	None				
4	<b>alamat</b>	text	utf8mb4_0900_ai_ci	No	None				
5	<b>no_telp</b>	varchar(13)	utf8mb4_0900_ai_ci	No	None				
6	<b>jurusan</b>	varchar(25)	utf8mb4_0900_ai_ci	No	None				
7	<b>kegiatan</b>	enum('kkn', 'kkp', 'none')	utf8mb4_0900_ai_ci	Yes	NULL				

- e. Membuat data table “**pendaftaran\_kkn**”

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id_pendaftaran</b>	int		No	None				
2	<b>id_mahasiswa</b>	int		No	None				
3	<b>id_tempat</b>	int		No	None				
4	<b>id_dosen</b>	int		No	None				
5	<b>tgl_daftar</b>	date		No	None				
6	<b>status</b>	varchar(15)	utf8mb4_0900_ai_ci	No	None				

f. Membuat data table “**pendaftaran\_kkp**”

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 <b>id_pendaftaran</b>	int			No	None			More
<input type="checkbox"/>	2 <b>id_mahasiswa</b>	int			No	None			More
<input type="checkbox"/>	3 <b>id_tempat</b>	int			No	None			More
<input type="checkbox"/>	4 <b>tgl_daftar</b>	date			No	None			More

Check all    With selected: Primary Unique

g. Membuat data table “**pendaftaran\_kkpdetail**”

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 <b>id_pendaftaran</b>	int			No	None			More
<input type="checkbox"/>	2 <b>id_dosen</b>	int			Yes	NULL			More
<input type="checkbox"/>	3 <b>status</b>	enum('waiting', 'approve', 'not approve')	utf8mb4_0900_ai_ci		Yes	NULL			More

Check all    With selected: Primary Unique Index Spatial Fulltext

h. Membuat data table “**pengguna**”

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 <b>id_pengguna</b>	int			No	None	AUTO_INCREMENT		More
<input type="checkbox"/>	2 <b>email</b>	varchar(50)	utf8mb4_0900_ai_ci		No	None			More
<input type="checkbox"/>	3 <b>password</b>	varchar(25)	utf8mb4_0900_ai_ci		No	None			More
<input type="checkbox"/>	4 <b>nama_pengguna</b>	varchar(50)	utf8mb4_0900_ai_ci		No	None			More
<input type="checkbox"/>	5 <b>level</b>	enum('admin', 'dosen', 'mahasiswa', '')	utf8mb4_0900_ai_ci		No	None			More

Check all    With selected: Primary Unique Index Spatial Fulltext

i. Membuat data table “**tempat\_kkn**”

<input type="checkbox"/>	1 <b>id_tempat</b>	int		No	None			More
<input type="checkbox"/>	2 <b>nama_tempat</b>	varchar(50)	utf8mb4_0900_ai_ci	No	None			More
<input type="checkbox"/>	3 <b>lokasi</b>	text	utf8mb4_0900_ai_ci	No	None			More
<input type="checkbox"/>	4 <b>program</b>	text	utf8mb4_0900_ai_ci	Yes	NULL			More
<input type="checkbox"/>	5 <b>id_dosen</b>	int		No	None			More
<input type="checkbox"/>	6 <b>jumlah_mahasiswa</b>	int		Yes	NULL			More

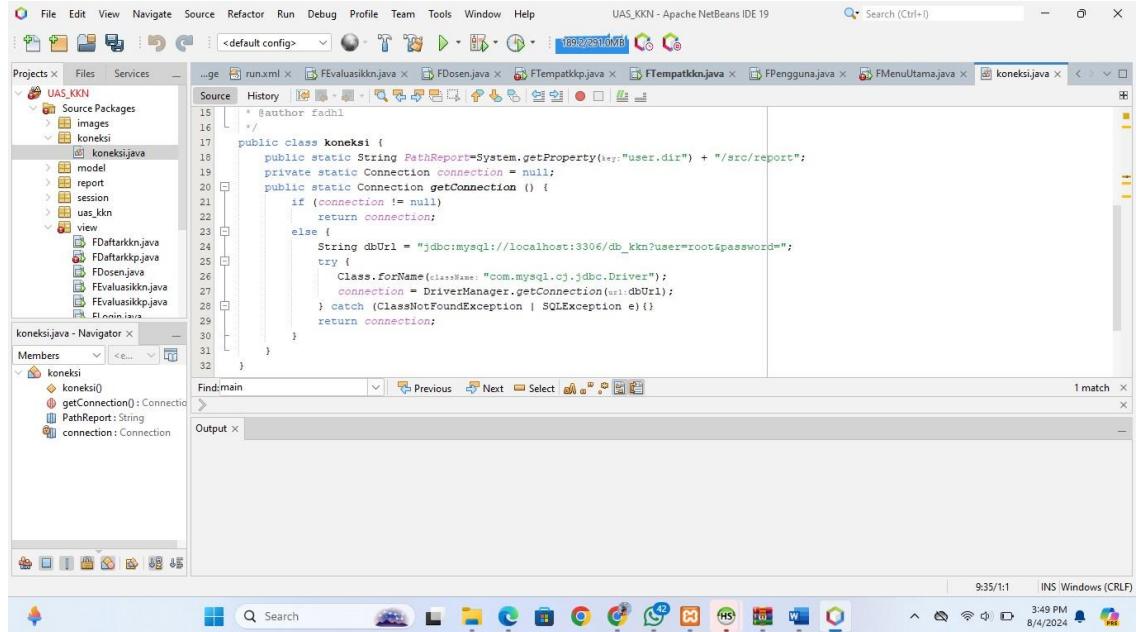
j. Membuat data table “**tempat\_kkp**”

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 <b>id_tempat</b>	int			No	None			More
<input type="checkbox"/>	2 <b>nama_perusahaan</b>	text	utf8mb4_0900_ai_ci		No	None			More
<input type="checkbox"/>	3 <b>alamat</b>	text	utf8mb4_0900_ai_ci		No	None			More
<input type="checkbox"/>	4 <b>nama_pembimbing</b>	varchar(50)	utf8mb4_0900_ai_ci		No	None			More
<input type="checkbox"/>	5 <b>notelp</b>	varchar(13)	utf8mb4_0900_ai_ci		No	None			More
<input type="checkbox"/>	6 <b>tgl_mulai</b>	date			No	None			More
<input type="checkbox"/>	7 <b>tgl_berakhir</b>	date			No	None			More

## BAB III

### MEMBUAT VIEW

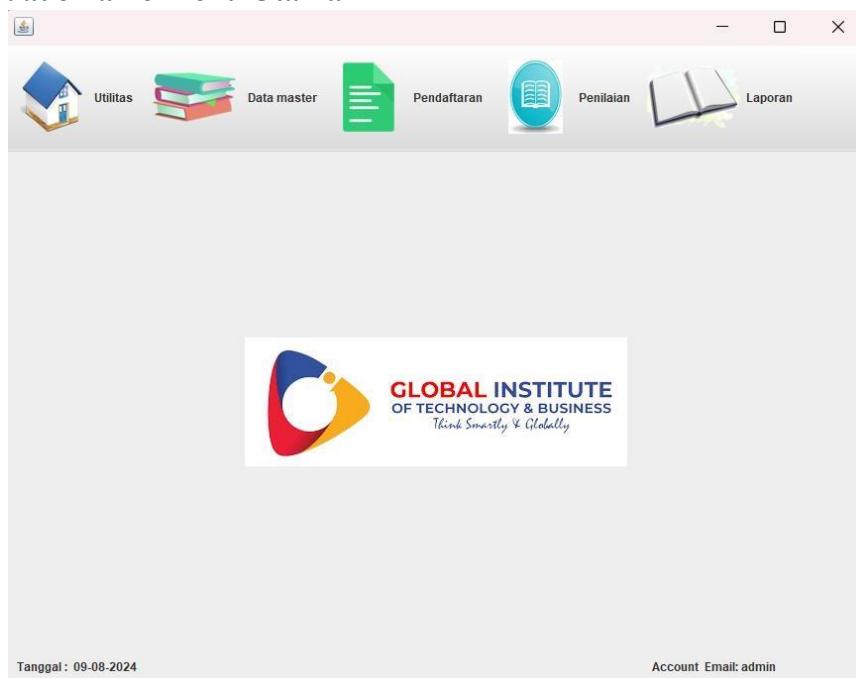
#### 3.1 Konfigurasi pada Database pada Apache Netbeans



Buat folder koneksi dan masukan sourcode ini untuk koneksi ke database

```
package koneksi; import  
java.sql.Connection; import  
java.sql.DriverManager; import  
java.sql.PreparedStatement;  
import java.sql.ResultSet; import  
java.sql.SQLException;  
import javax.swing.JOptionPane;  
  
public class koneksi {  
    public static String PathReport=System.getProperty("user.dir") +  
    "/src/report";    private static Connection connection = null;    public static  
    Connection getConnection () {  
        if (connection != null)  
            return connection;        else {  
            String dbUrl = "jdbc:mysql://localhost:3306/db_kkn?user=root&password=";  
            try {  
                Class.forName("com.mysql.cj.jdbc.Driver");  
                connection = DriverManager.getConnection(dbUrl);            }  
                catch (ClassNotFoundException | SQLException e){}  
            return connection;  
        }  
    } }
```

### 3.2 Buat Jframe Menu Utama



Berikut sourcecode nya

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package view;
import javax.swing.JFrame;
import session.session;
import java.io.File;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
import java.util.Map;
import javax.swing.JOptionPane;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.engine.design.JasperDesign;
import net.sf.jasperreports.engine.xml.JRXmlLoader;
import net.sf.jasperreports.view.JasperViewer;
import org.apache.log4j.Logger;
import net.sf.jasperreports.engine.JRException;
import java.sql.Connection;
```

```
/**
```

```

*
* @author fadhl
*/
public class FMenuUtama extends javax.swing.JFrame {
    private static final Logger logger = Logger.getLogger(FMenuUtama.class);
    public static String levelUser;

    //Deklarasi Variabel
    private Map<String, Object> param = new HashMap<>();

    //Deklarasi Variabel utk Laporan
    JasperReport jasRep;
    JasperPrint jasPrint;
    JasperDesign jasDes;
    /**
     * Creates new form FMenuUtama
     */
    public FMenuUtama() {
        initComponents();
        loginGagal();
        //method tanggal
        setTanggal();
        setMenuAkses();
    }

    public void setTanggal() {
        // Mendapatkan tanggal saat ini
        LocalDate today = LocalDate.now();

        // Menentukan format tanggal
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");

        // Mengonversi tanggal ke format yang diinginkan
        String formattedDate = today.format(formatter);

        // Menetapkan tanggal yang diformat ke komponen
        ITanggal.setText(formattedDate);
    }

    public static void loginGagal () {
        mnPendaftaran.setVisible(false);
        mnData.setVisible(false);
        mnLaporan.setVisible(false);
        mnPenilaian.setVisible(false);
        smLogin.setText("Login");
    }

    public static void loginSukses() {
        // Akses Admin
        mnPendaftaran.setVisible(true);
        mnData.setVisible(true);
    }
}

```

```

mnLaporan.setVisible(true);
mnPenilaian.setVisible(true);
smLapMhs.setVisible(true);
smLapkkn.setVisible(true);
smLapkkp.setVisible(true);
smLapNilaikkp.setVisible(true);
smLapNilaikkn.setVisible(true);
smKKN.setVisible(true);

// Ambil email dari sesi dan tampilkan di lblAkun
String email = session.getInstance().getUsername();
if (lblAkun != null) {
    lblAkun.setText("Email: " + email); // Set label dengan email pengguna
}

if (levelUser != null) {
    switch (levelUser) {
        case "admin":
            // Akses admin
            break;

        case "dosen":
            // Pengecekan status dosen
            String statusDosen = session.getInstance().getStatus();

            if ("pengaji/penilai".equals(statusDosen)) {
                mnPendaftaran.setVisible(false);
                mnData.setVisible(false);
                mnLaporan.setVisible(true);
                mnPenilaian.setVisible(true);
                smPenilaiankkp.setVisible(true);
                smPenilaiankkn.setVisible(true);
                smLapMhs.setVisible(true);
                smLapkkn.setVisible(true);
                smLapkkp.setVisible(true);
                smLapNilaikkp.setVisible(true);
                smLapNilaikkn.setVisible(true);
            } else if ("kkp".equals(statusDosen) || "kkn".equals(statusDosen)) {
                mnPendaftaran.setVisible(false);
                mnData.setVisible(false);
                mnLaporan.setVisible(true);
                mnPenilaian.setVisible(true);
                smLapMhs.setVisible(true);
                smLapkkn.setVisible(true);
                smLapkkp.setVisible(true);
                smLapNilaikkp.setVisible(true);
                smLapNilaikkn.setVisible(true);
                smPenilaiankkp.setVisible(true);
                smPenilaiankkn.setVisible(true);
            } else {
                // Status dosen tidak dikenali
            }
    }
}

```

```

        loginGagal();
    }
    break;

case "mahasiswa":
    // Akses mahasiswa
    String kegiatanMahasiswa = session.getInstance().getKegiatan();

    if ("kkn".equals(kegiatanMahasiswa)) {
        mnData.setVisible(false);
        mnPendaftaran.setVisible(true);
        smLapMhs.setVisible(false);
        mnPenilaian.setVisible(false);
        mnLaporan.setVisible(true);
        // Hanya tampilkan submenu KKN
        smKKN.setVisible(true); // Pastikan submenu ini ada
        smKKP.setVisible(false);
        smLapMhs.setVisible(true);
        smLapkkn.setVisible(true);
        smLapkkp.setVisible(true);
        smLapNilaikkp.setVisible(true);
        smLapNilaikkn.setVisible(true);
    } else if ("kkp".equals(kegiatanMahasiswa)) {
        mnData.setVisible(false);
        mnPendaftaran.setVisible(true);
        smLapMhs.setVisible(false);
        mnPenilaian.setVisible(false);
        mnLaporan.setVisible(true);
        // Hanya tampilkan submenu KKP
        smKKP.setVisible(true); // Pastikan submenu ini ada
        smKKN.setVisible(false);
        smLapMhs.setVisible(true);
        smLapkkn.setVisible(true);
        smLapkkp.setVisible(true);
        smLapNilaikkp.setVisible(true);
        smLapNilaikkn.setVisible(true);
    } else {
        // Kegiatan mahasiswa tidak dikenali
        loginGagal();
    }
    break;

default:
    // Jika levelUser tidak dikenali, sembunyikan semua menu
    loginGagal();
    break;
}
} else {
    loginGagal();
}

```

```

        smLogin.setText("Logout");
    }

    private void setMenuAkses() {
        if (session.getInstance().getUsername() != null) {
            loginSukses();
        } else {
            loginGagal();
        }
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        lTanggal = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        lblAkun = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jMenuBar1 = new javax.swing.JMenuBar();
        mnUtilitas = new javax.swing.JMenu();
        smLogin = new javax.swing.JMenuItem();
        smKeluar = new javax.swing.JMenuItem();
        mnData = new javax.swing.JMenu();
        smPengguna = new javax.swing.JMenuItem();
        smMahasiswa = new javax.swing.JMenuItem();
        smDosen = new javax.swing.JMenuItem();
        smTempatKKN = new javax.swing.JMenuItem();
        jMenuItem1 = new javax.swing.JMenuItem();
        mnPendaftaran = new javax.swing.JMenu();
        smKKN = new javax.swing.JMenuItem();
        smKKP = new javax.swing.JMenuItem();
        mnPenilaian = new javax.swing.JMenu();
        smPenilaiankkn = new javax.swing.JMenuItem();
        smPenilaiankkp = new javax.swing.JMenuItem();
        mnLaporan = new javax.swing.JMenu();
        smLapMhs = new javax.swing.JMenuItem();
        smLapkkn = new javax.swing.JMenuItem();
        smLapkkp = new javax.swing.JMenuItem();
        smLapNilaikkn = new javax.swing.JMenuItem();
        smLapNilaikkp = new javax.swing.JMenuItem();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setText("Tanggal :");
    }

```

```

lTanggal.setText("dd/mm/yyyy");

jLabel2.setText("Account");

lblAkun.setText("...");

jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/LgoGlobal.png"))); // NOI18N

mnUtilitas.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/kelas.png"))); // NOI18N
mnUtilitas.setText("Utilitas");

smLogin.setText("Login");
smLogin.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        smLoginActionPerformed(evt);
    }
});
mnUtilitas.add(smLogin);

smKeluar.setText("Keluar");
smKeluar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        smKeluarActionPerformed(evt);
    }
});
mnUtilitas.add(smKeluar);

jMenuBar1.add(mnUtilitas);

mnData.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/duction book.png"))); // NOI18N
mnData.setText("Data master");

smPengguna.setText("Data Pengguna");
smPengguna.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        smPenggunaActionPerformed(evt);
    }
});
mnData.add(smPengguna);

smMahasiswa.setText("Data Mahasiswa");
smMahasiswa.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        smMahasiswaActionPerformed(evt);
    }
});

```

```

    });
    mnData.add(smMahasiswa);

    smDosen.setText("Data Dosen");
    smDosen.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            smDosenActionPerformed(evt);
        }
    });
    mnData.add(smDosen);

    smTempatKKN.setText("Data Tempat KKN");
    smTempatKKN.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            smTempatKKNActionPerformed(evt);
        }
    });
    mnData.add(smTempatKKN);

    jMenuItem1.setText("Data Tempat KKP");
    jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jMenuItem1ActionPerformed(evt);
        }
    });
    mnData.add(jMenuItem1);

    jMenuBar1.add(mnData);

    mnPendaftaran.setIcon(new
        javax.swing.ImageIcon(getClass().getResource("/images/nilai1.png"))); // NOI18N
    mnPendaftaran.setText("Pendaftaran");

    smKKN.setText("KKN");
    smKKN.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            smKKNActionPerformed(evt);
        }
    });
    mnPendaftaran.add(smKKN);

    smKKP.setText("KKP");
    smKKP.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            smKKPActionPerformed(evt);
        }
    });
    mnPendaftaran.add(smKKP);

    jMenuBar1.add(mnPendaftaran);

```

```

mnPenilaian.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/matkul.png"))); // NOI18N
mnPenilaian.setText("Penilaian");

smPenilaiankkn.setText("Penilaian KKN");
smPenilaiankkn.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        smPenilaiankknMouseClicked(evt);
    }
});
smPenilaiankkn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        smPenilaiankknActionPerformed(evt);
    }
});
mnPenilaian.add(smPenilaiankkn);

smPenilaiankp.setText("Penilaian KKP");
smPenilaiankp.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        smPenilaiankpMouseClicked(evt);
    }
});
smPenilaiankp.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        smPenilaiankpActionPerformed(evt);
    }
});
mnPenilaian.add(smPenilaiankp);

jMenuBar1.add(mnPenilaian);

mnLaporan.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/book1.png"))); // NOI18N
mnLaporan.setText("Laporan");

smLapMhs.setText("Laporan Mahasiswa ");
smLapMhs.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        smLapMhsActionPerformed(evt);
    }
});
mnLaporan.add(smLapMhs);

smLapkkn.setText("Laporan Mahasiswa KKN");
smLapkkn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        smLapkknActionPerformed(evt);
    }
});
mnLaporan.add(smLapkkn);

```

```
smLapkkp.setText("Laporan Mahasiswa KKP");
smLapkkp.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        smLapkkpActionPerformed(evt);
    }
});
mnLaporan.add(smLapkkp);

smLapNilaikkn.setText("Laporan Nilai KKN");
smLapNilaikkn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        smLapNilaikknActionPerformed(evt);
    }
});
mnLaporan.add(smLapNilaikkn);

smLapNilaikkp.setText("Laporan Nilai KKP");
smLapNilaikkp.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        smLapNilaikkpActionPerformed(evt);
    }
});
mnLaporan.add(smLapNilaikkp);

jMenuBar1.add(mnLaporan);

setJMenuBar(jMenuBar1);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel1)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(lTanggal)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
            javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jLabel2)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(lblAkun, javax.swing.GroupLayout.PREFERRED_SIZE, 153,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap()
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup()
                .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 871,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(layout.createSequentialGroup()
                    .addGap(0, 0, Short.MAX_VALUE)))
            .addContainerGap())
    );
);
layout.setVerticalGroup(
```

```

        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 506,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(10, 10, 10)
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(10, 10, 10)
            .addComponent(lTanggal, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(10, 10, 10)
            .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(10, 10, 10)
            .addComponent(lblAkun, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(10, 10, 10)
            .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(10, 10, 10)
            .addComponent(lblKet, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
                javax.swing.GroupLayout.PREFERRED_SIZE)
        )
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel1)
            .addComponent(lTanggal)
            .addComponent(jLabel2)
            .addComponent(lblAkun)
        )
    );
}

pack();
setLocationRelativeTo(null);
}// </editor-fold>

private void smLoginActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    // TODO add your handling code here:
    if ("Login".equals(FMenuUtama.smLogin.getText())) {
        FLogin login = new FLogin();
        login.setVisible(true);
    }
    else {
        FMenuUtama.loginGagal();
    }
}

private void smKKPActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FDaftarkkp kkp = new FDaftarkkp();
    kkp.setVisible(true);
    DesktopPane.add(eval);
}

private void smDosenActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FDosen dsn = new FDosen();
    dsn.setVisible(true);
    DesktopPane.add(dsn);
}

private void smLapkknActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JPrmkkn rkkn = new JPrmkkn();
    rkkn.setVisible(true);
}

```

```

private void smMahasiswaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FMahasiswa mhs = new FMahasiswa();
    mhs.setVisible(true);
//    DesktopPane.add(mhs);
}

private void smTempatKKNActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FTempatkkn kkn = new FTempatkkn();
    kkn.setVisible(true);
//    DesktopPane.add(kkn);
}

private void smPenggunaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FPengguna pgn = new FPengguna();
    pgn.setVisible(true);
//    DesktopPane.add(pgn);
}

private void smKKNActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FDaftarkkn dft = new FDaftarkkn();
    dft.setVisible(true);
//    DesktopPane.add(dft);
}

private void smLapMhsActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        // Pastikan file laporan ada
        File report = new File("src/report/repMhs.jrxml");
        if (!report.exists()) {
            logger.error("File laporan tidak ditemukan: " + report.getAbsolutePath());
            JOptionPane.showMessageDialog(null, "File laporan tidak ditemukan", "Cetak
Laporan",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        // Muat, kompilasi, dan isi laporan
        JasperDesign jasDes = JRXmlLoader.load(report);
        JasperReport jasRep = JasperCompileManager.compileReport(jasDes);
        // Gunakan koneksi yang sudah ada
        Connection connection = koneksi.koneksi.getConnection();
        if (connection != null && !connection.isClosed()) {
            JasperPrint jasPrint = JasperFillManager.fillReport(jasRep, null, connection);

            // Tampilkan laporan
        }
    }
}

```

```

        JasperViewer.viewReport(jasPrint, false);
    } else {
        logger.error("Connection is closed.");
        JOptionPane.showMessageDialog(null, "Koneksi database tidak tersedia",
        "Cetak Laporan",
        JOptionPane.ERROR_MESSAGE);
    }
} catch (JRException e) {
    // Log exception dan tampilkan pesan kesalahan
    logger.error("Gagal memproses laporan JasperReports", e);
    JOptionPane.showMessageDialog(null, "Gagal Membuka Laporan", "Cetak
Laporan",
    JOptionPane.ERROR_MESSAGE);
} catch (Exception e) {
    // Tangani exception umum
    logger.error("Terjadi kesalahan", e);
    JOptionPane.showMessageDialog(null, "Terjadi kesalahan", "Cetak Laporan",
    JOptionPane.ERROR_MESSAGE);
}

}

private void smKeluarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    dispose();
}

private void smPenilaiankknMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
}

private void smPenilaiankkpMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
}

private void smPenilaiankknActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FEvaluasikkn Ekn = new FEvaluasikkn();
    Ekn.setVisible(true);
}

private void smPenilaiankkpActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FEvaluasikkp kkp = new FEvaluasikkp();
    kkp.setVisible(true);
}

private void smLapkkpActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

```

```

JPrmkkp kkp = new JPrmkkp();
kkp.setVisible(true);
}

private void smLapNilaikknActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JPrmNilaikkn rkkn = new JPrmNilaikkn();
    rkkn.setVisible(true);
}

private void smLapNilaikkpActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JPrmNilaikkp nkpp = new JPrmNilaikkp();
    nkpp.setVisible(true);
}

private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FTempatkkp tkkp = new FTempatkkp();
    tkkp.setVisible(true);
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
    feel.
        * For details see
    http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
            javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

        java.util.logging.Logger.getLogger(FMenuUtama.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

        java.util.logging.Logger.getLogger(FMenuUtama.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
}

```

```

java.util.logging.Logger.getLogger(FMenuUtama.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
    java.util.logging.Logger.getLogger(FMenuUtama.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new FMenuUtama().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JMenuItem jMenuItem1;
private javax.swing.JLabel lTanggal;
private static javax.swing.JLabel lblAkun;
private static javax.swing.JMenu mnData;
private static javax.swing.JMenu mnLaporan;
private static javax.swing.JMenu mnPendaftaran;
private static javax.swing.JMenu mnPenilaian;
private javax.swing.JMenu mnUtilitas;
private static javax.swing.JMenuItem smDosen;
private static javax.swing.JMenuItem smKKN;
private static javax.swing.JMenuItem smKKP;
private javax.swing.JMenuItem smKeluar;
private static javax.swing.JMenuItem smLapMhs;
private static javax.swing.JMenuItem smLapNilaikkn;
private static javax.swing.JMenuItem smLapNilaikkp;
private static javax.swing.JMenuItem smLapkkn;
private static javax.swing.JMenuItem smLapkkp;
private static javax.swing.JMenuItem smLogin;
private static javax.swing.JMenuItem smMahasiswa;
private static javax.swing.JMenuItem smPengguna;
private static javax.swing.JMenuItem smPenilaiankkn;
private static javax.swing.JMenuItem smPenilaiankkp;
private static javax.swing.JMenuItem smTempatKKN;
// End of variables declaration
}

```

### 3.3 Frame login



```
package view;

import koneksi.koneksi;
import session.session;
import java.sql.Connection;
import java.sql.PreparedStatement;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
 *
 * @author Fadhlansyah
 */
public class FLogin extends javax.swing.JFrame {

    private final Connection dbConnection;
    private PreparedStatement pstmt;
    private ResultSet rs;

    /**
     * Creates new form FrameLogin
     */
    public FLogin() {
        initComponents();
        dbConnection = koneksi.getConnection();
        tUser.requestFocus();
    }
}
```

Diatas merupakan method yg digunakan pada Frame Login (FLogin), yg kemudian akan digunakan pada button dengan Action Performed:

```
private void bLoginActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    // Query untuk memeriksa kredensial pengguna  
    String displayQuery = "SELECT * FROM pengguna WHERE email=? AND  
password=?";  
    String user = tUser.getText().trim();  
    char[] passArray = tPass.getPassword(); // Menggunakan JPasswordField  
    String pass = new String(passArray).trim();  
  
    // Validasi input  
    if (user.isEmpty() || pass.isEmpty()) {  
        JOptionPane.showMessageDialog(rootPane, "Email dan password harus diisi");  
        return;  
    }  
  
    try (PreparedStatement pstmt = dbConnection.prepareStatement(displayQuery)) {  
        pstmt.setString(1, user);  
        pstmt.setString(2, pass);  
  
        try (ResultSet rs = pstmt.executeQuery()) {  
            if (rs.next()) {  
                String levelUser = rs.getString("level");  
                String namaPengguna = rs.getString("nama_pengguna");  
  
                // Menampilkan pesan login berhasil  
                JOptionPane.showMessageDialog(rootPane, "User " + user + " berhasil  
login");  
                session.getInstance().setUsername(user);  
                session.getInstance().setLevelUser(levelUser);  
                FMenuUtama.levelUser = levelUser;  
  
                // Pengecekan khusus untuk mahasiswa  
                if ("mahasiswa".equalsIgnoreCase(levelUser)) {  
                    String kegiatanQuery = "SELECT kegiatan FROM mahasiswa WHERE  
nama=?";  
                    try (PreparedStatement pstmtKegiatan =  
dbConnection.prepareStatement(kegiatanQuery)) {  
                        pstmtKegiatan.setString(1, namaPengguna);  
  
                        try (ResultSet rsKegiatan = pstmtKegiatan.executeQuery()) {  
                            if (rsKegiatan.next()) {  
                                String kegiatanMahasiswa = rsKegiatan.getString("kegiatan");  
                                session.getInstance().setKegiatan(kegiatanMahasiswa);  
                            } else {  
                                // Handle case where no activity found  
                            }  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

```

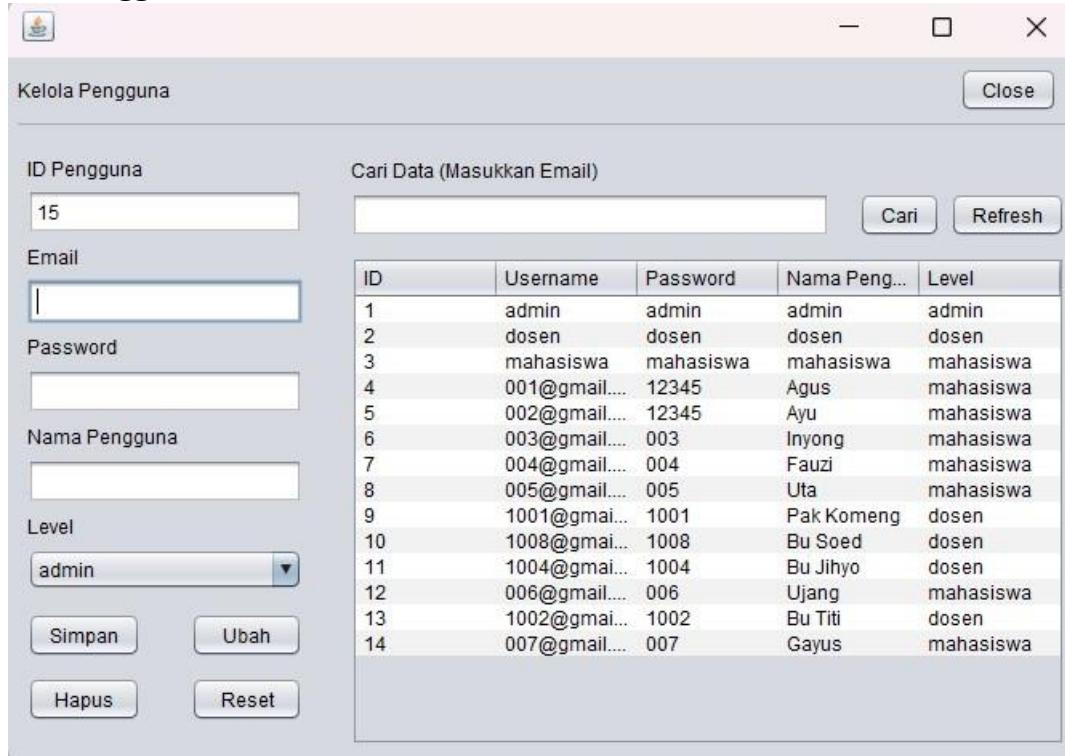
        session.getInstance().setKegiatan("unknown"); // Kegiatan tidak
ditemukan
    }
}
}
}
}
// Pengecekan khusus untuk dosen
else if ("dosen".equalsIgnoreCase(levelUser)) {
    String statusQuery = "SELECT status FROM dosen WHERE
nama_dosen=?";
    try (PreparedStatement pstmtStatus =
dbConnection.prepareStatement(statusQuery)) {
        pstmtStatus.setString(1, namaPengguna);

        try (ResultSet rsStatus = pstmtStatus.executeQuery()) {
            if (rsStatus.next()) {
                String statusDosen = rsStatus.getString("status");
                session.getInstance().setStatus(statusDosen);
            } else {
                session.getInstance().setStatus("unknown"); // Status tidak
ditemukan
            }
        }
    }
}

// Panggil metode untuk memperbarui akses menu
FMenuUtama.loginSukses();
} else {
    JOptionPane.showMessageDialog(rootPane, "User " + user + " gagal
login\nCek lagi user dan password anda");
    FMenuUtama.loginGagal();
}
}
} catch (SQLException e) {
    JOptionPane.showMessageDialog(rootPane, "Terjadi kesalahan saat login: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
} finally {
    this.dispose();
}
}

```

### 3.4 Frame Pengguna



Berikut source code frame pengguna :

```
package view;
import koneksi.koneksi;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.Arrays;

/**
 *
 * @author Fadhlhan
 */
public class FPengguna extends javax.swing.JFrame {

    Connection connection;
    DefaultTableModel model;
    /**
     * Creates new form FramePengguna
     */
    public FPengguna() {
```

```

initComponents();
connection = koneksi.getConnection();
tEmail.requestFocus();
getDataTable();
getcmbLevel();
generateAutoNumber();
}

private void getDataTable() {
model = (DefaultTableModel) table.getModel();
model.setRowCount(0);
try{
    Statement stat = connection.createStatement();
    String sql = "SELECT * FROM pengguna";
    ResultSet res = stat.executeQuery(sql);
    while (res.next()){
        Object[] obj = new Object [5];
        obj [0] = res.getString("id_pengguna");
        obj [1] = res.getString("email");
        obj [2] = res.getString("password");
        obj [3] = res.getString("nama_pengguna");
        obj [4] = res.getString("level");
        model.addRow(obj);
    }
} catch (SQLException err) {
    err.printStackTrace();
}
}

private void getcmbLevel(){
cmbLevel.removeAllItems();
try{
    Statement stat = connection.createStatement();
    String sql = "SELECT * FROM pengguna";
    ResultSet res = stat.executeQuery(sql);
    while(res.next()){
        cmbLevel.addItem(res.getString("level"));
    }
} catch(SQLException err){
    err.printStackTrace();
}
}

private void generateAutoNumber() {
try {
    Statement stat = connection.createStatement();
    String sql = "SELECT MAX(id_pengguna) AS max_id FROM pengguna";
    ResultSet res = stat.executeQuery(sql);

```

```

        if (res.next()) {
            int maxId = res.getInt("max_id");
            int newId = maxId + 1;
            tID.setText(String.valueOf(newId));
        } else {
            // Handle the case where no records exist yet
            tID.setText("1");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void refresh() {
    model = (DefaultTableModel) table.getModel();
    model.setRowCount(0);
    getDataTable();
    generateAutoNumber();
}

private void reset() {
    tID.setText("");
    tEmail.setText("");
    tPass.setText("");
    tNama.setText("");
    tEmail.setEditable(true);
    bSimpan.setEnabled(true);
}

private void insert() {
    String sql = "INSERT INTO pengguna (id_pengguna, email, password,
    nama_pengguna, level) VALUES (?, ?, ?, ?, ?)";

    // Ambil password dari JPasswordField
    char[] passArray = tPass.getPassword(); // Menggunakan JPasswordField
    String pass = new String(passArray); // Konversi char[] ke String

    // Validasi input
    if (tID.getText().trim().isEmpty() || tEmail.getText().trim().isEmpty() ||
    pass.trim().isEmpty() || tNama.getText().trim().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Semua field harus diisi", "Error",
        JOptionPane.ERROR_MESSAGE);
        return;
    }

    try (PreparedStatement statement = connection.prepareStatement(sql,
    Statement.RETURN_GENERATED_KEYS)) {
        // Set parameter untuk PreparedStatement
        statement.setString(1, tID.getText().trim());

```

```

statement.setString(2, tEmail.getText().trim());
statement.setString(3, pass.trim());
statement.setString(4, tNama.getText().trim());
statement.setString(5, cmbLevel.getSelectedItem().toString().trim());

// Eksekusi query
statement.executeUpdate();

// Informasikan pengguna bahwa data berhasil disimpan
JOptionPane.showMessageDialog(this, "Data berhasil disimpan", "Informasi",
JOptionPane.INFORMATION_MESSAGE);

// Menghapus password dari memori
Arrays.fill(passArray, '0'); // Menghapus data password dari memori
} catch (SQLException ex) {
ex.printStackTrace();
JOptionPane.showMessageDialog(this, "Terjadi kesalahan saat menyimpan
data: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
}
}

private void update() {
String sql = "UPDATE pengguna SET email=?, password=?, nama_pengguna=?,
level=? WHERE id_pengguna=?";

// Ambil password dari JPasswordField
char[] passArray = tPass.getPassword(); // Menggunakan JPasswordField
String pass = new String(passArray); // Konversi char[] ke String

// Validasi input
if (tEmail.getText().trim().isEmpty() || pass.trim().isEmpty() ||
tNama.getText().trim().isEmpty() || tID.getText().trim().isEmpty()) {
JOptionPane.showMessageDialog(this, "Semua field harus diisi", "Error",
JOptionPane.ERROR_MESSAGE);
Arrays.fill(passArray, '0'); // Menghapus data password dari memori
return;
}

try (PreparedStatement statement = connection.prepareStatement(sql)) {
// Set parameter untuk PreparedStatement
statement.setString(1, tEmail.getText().trim());
statement.setString(2, pass.trim());
statement.setString(3, tNama.getText().trim());
statement.setString(4, cmbLevel.getSelectedItem().toString().trim());
statement.setString(5, tID.getText().trim());

// Eksekusi query
int rowsUpdated = statement.executeUpdate();
}
}

```

```

        if (rowsUpdated > 0) {
            JOptionPane.showMessageDialog(this, "Data berhasil diperbarui",
                "Informasi", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this, "Data tidak ditemukan untuk
diperbarui", "Informasi", JOptionPane.INFORMATION_MESSAGE);
        }

        // Menghapus password dari memori
        Arrays.fill(passArray, '0'); // Menghapus data password dari memori
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Terjadi kesalahan saat memperbarui
data: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}

private void delete() {
    PreparedStatement statement = null;
    String sql = "DELETE FROM pengguna WHERE id_pengguna=?";
    try {
        statement = connection.prepareStatement(sql);
        statement.setString(1, tID.getText());
        statement.executeUpdate();
    } catch (SQLException ex) {
        ex.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

private void search() {
    model = (DefaultTableModel) table.getModel();
    PreparedStatement statement = null;
    try{
        String sql = "SELECT * FROM pengguna WHERE email like ?";
        statement = connection.prepareStatement(sql);
        statement.setString(1, "%" + tCari.getText() + "%");
        ResultSet res = statement.executeQuery();
        while(res.next()) {
            Object[] obj = new Object[5];
            obj[0] = res.getString("id_pengguna");
            obj[1] = res.getString("email");
            obj[2] = res.getString("password");
        }
    }
}

```

```

        obj[3] = res.getString("nama_pengguna");
        obj[4] = res.getString("level");
        model.addRow(obj);
    }
} catch (SQLException err) {
    err.printStackTrace();
}
}

private void bSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    char[] passArray = tPass.getPassword();
    String pass = new String(passArray);

    // Validasi input
    if (!tID.getText().trim().isEmpty() &&
        !tEmail.getText().trim().isEmpty() &&
        !pass.trim().isEmpty() &&
        !tNama.getText().trim().isEmpty()) {

        // Panggil metode insert untuk menyimpan data
        insert();

        // Refresh dan reset form
        refresh();
        reset();

        // Tampilkan notifikasi berhasil
        JOptionPane.showMessageDialog(this,
            "Data User/Pengguna berhasil ditambahkan",
            "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
    } else {
        // Tampilkan notifikasi jika ada field yang kosong
        JOptionPane.showMessageDialog(this,
            "Lengkapi Form Terlebih Dahulu!",
            "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }

    // Hapus password dari memori
    Arrays.fill(passArray, '0');
}

private void bRefreshActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    refresh();
}

private void bResetActionPerformed(java.awt.event.ActionEvent evt) {

```

```

// TODO add your handling code here:
reset();
}

private void tableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    tID.setText(table.getModel().getValueAt(table.getSelectedRow(),
0).toString());
    tEmail.setText(table.getModel().getValueAt(table.getSelectedRow(),
1).toString());
    tPass.setText(table.getModel().getValueAt(table.getSelectedRow(),
2).toString());
    tNama.setText(table.getModel().getValueAt(table.getSelectedRow(),
3).toString());

    cmbLevel.setSelectedItem(table.getModel().getValueAt(table.getSelectedRow(),
4).toString());
    tID.setEditable(true);
    bSimpan.setEnabled(false);
}

private void bUbahActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    char[] passArray = tPass.getPassword();
    String pass = new String(passArray);

    // Validasi input
    if (!tID.getText().trim().isEmpty()) {
        if (!tEmail.getText().trim().isEmpty() &&
            !pass.trim().isEmpty() &&
            !tNama.getText().trim().isEmpty() &&
            !cmbLevel.getSelectedItem().toString().trim().isEmpty()) {

            // Panggil metode update untuk memperbarui data
            update();

            // Refresh dan reset form
            refresh();
            reset();

            // Tampilkan notifikasi berhasil
            JOptionPane.showMessageDialog(this,
                "Data pengguna berhasil diubah",
                "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
    } else {
        // Tampilkan notifikasi jika ada field yang kosong
        JOptionPane.showMessageDialog(this,
            "Lengkapi form terlebih dahulu",

```

```

        "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
} else {
    // Tampilkan notifikasi jika ID kosong
    JOptionPane.showMessageDialog(this,
        "Pilih Data terlebih dahulu!",
        "Notifikasi", JOptionPane.WARNING_MESSAGE);
}

// Hapus password dari memori
Arrays.fill(passArray, '0');
}

private void bHapusActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(!tID.getText().trim().isEmpty()) {
        int alert = JOptionPane.showConfirmDialog(this,
            "Anda yakin ingin menghapus Pengguna ini?",
            "Notifikasi", JOptionPane.YES_NO_OPTION,
            JOptionPane.QUESTION_MESSAGE);
        if(alert == JOptionPane.YES_OPTION) {
            delete();
            refresh();
            reset();
            JOptionPane.showMessageDialog(this,
                "Data Pengguna berhasil dihapus",
                "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(this,
            "Pilih data terlebih dahulu!",
            "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    dispose();
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    dispose();
}

private void bCariActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    model = (DefaultTableModel) table.getModel();
}

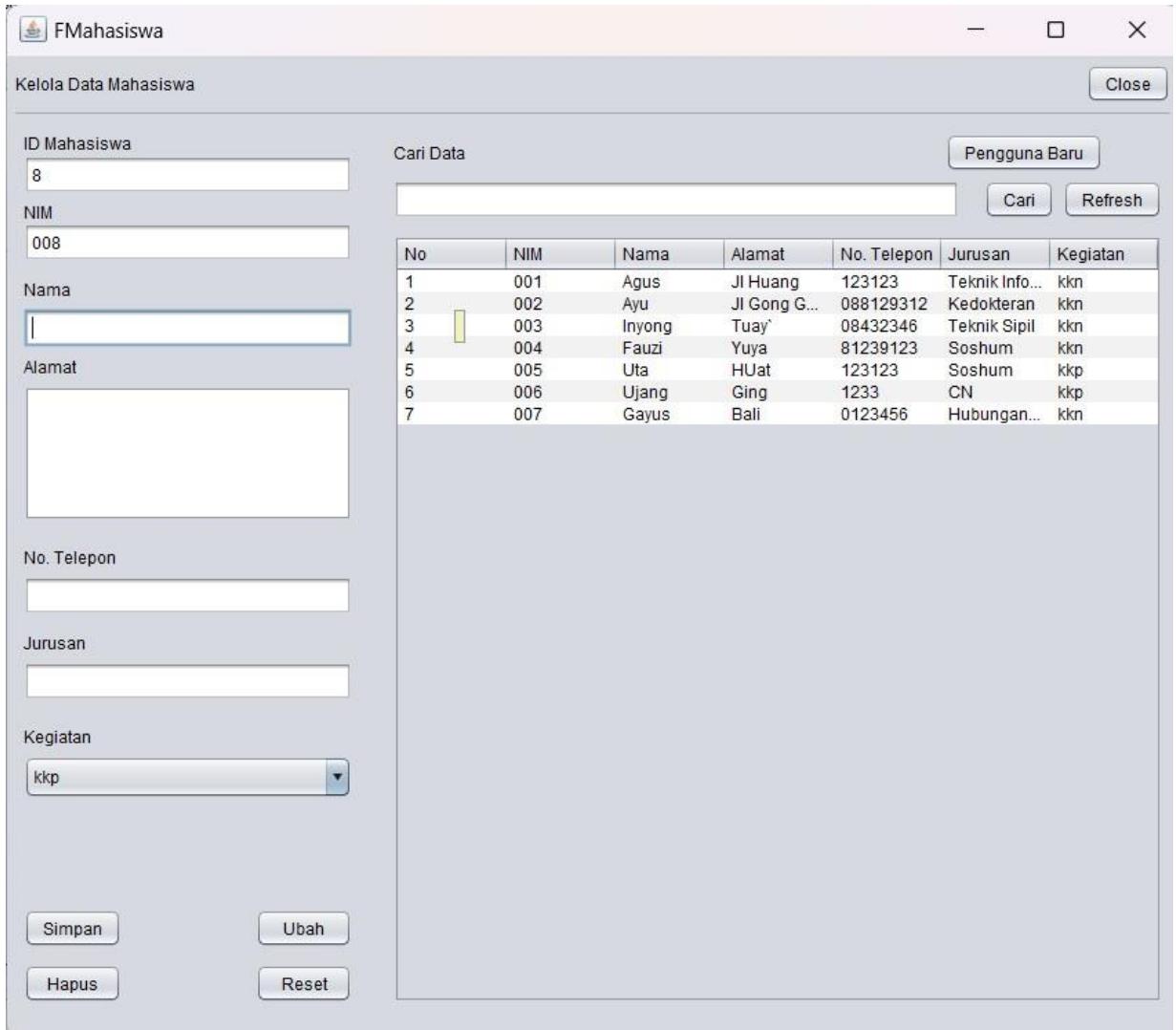
```

```

        model.setRowCount(0);
        search();
    }
}

```

### 3.5 Frame mahasiswa



Berikut source code mahasiswa :

```

package view;
import koneksi.koneksi;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.util.logging.Level;
import java.util.logging.Logger;

```

```
/**
```

```

*
* @author Fadhlhan
*/
public class FMahasiswa extends javax.swing.JFrame {
    Connection connection;
    DefaultTableModel model;
    /**
     * Creates new form FrameSiswa
     */
    public FMahasiswa() {
        initComponents();
        connection = koneksi.getConnection();
        getDataTable();
        // getcbKegiatan();
        AutoIDMhs();
        AutoNIM();
        tNama.requestFocus();
        refresh();
        tID.setEditable(false);
    }

    private void getDataTable(){
        model = (DefaultTableModel) table.getModel();
        model.setRowCount(0);
        try{
            Statement stat = connection.createStatement();
            String sql = "SELECT * FROM mahasiswa WHERE id_mahasiswa ";
            ResultSet res = stat.executeQuery(sql);
            while(res.next()){
                Object[ ] obj = new Object[7];
                obj[0] = res.getString("id_mahasiswa");
                obj[1] = res.getString("nim");
                obj[2] = res.getString("nama");
                obj[3] = res.getString("alamat");
                obj[4] = res.getString("no_telp");
                obj[5] = res.getString("jurusan");
                obj[6] = res.getString("kegiatan");
                model.addRow(obj);
            }
        }catch(SQLException err){
            err.printStackTrace();
        }
    }

    private void AutoIDMhs() {
        String sql = "SELECT MAX(id_mahasiswa) AS max_id FROM mahasiswa";
        try (Statement stat = connection.createStatement();
             ResultSet res = stat.executeQuery(sql)) {

```

```

int newId;
if (res.next()) {
    int maxId = res.getInt("max_id");
    newId = (maxId == 0) ? 1 : maxId + 1;
} else {
    // Handle the case where no records exist yet
    newId = 1;
}

// Convert the ID to a string without leading zeros
String formattedId = Integer.toString(newId);
tID.setText(formattedId);

} catch (SQLException e) {
    e.printStackTrace();
}
}

private void AutoNIM() {
    String sql = "SELECT MAX(nim) AS max_id FROM mahasiswa";
    try (Statement stat = connection.createStatement());
        ResultSet res = stat.executeQuery(sql)) {

    if (res.next()) {
        String maxId = res.getString("max_id");

        if (maxId == null) {
            // No records in the table
            tNim.setText("1");
        } else {
            // Extract the numeric part and increment it
            int maxNumericId = Integer.parseInt(maxId);
            int newId = maxNumericId + 1;

            // Format new ID with leading zeros
            String formattedId = String.format("%03d", newId);
            tNim.setText(formattedId);
        }
    } else {
        // Handle the case where the result set is empty
        tNim.setText("001");
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

```

```

private void getcbKegiatan(){
//    cbKegiatan.removeAllItems();
    try{
        Statement stat = connection.createStatement();
        String sql = "SELECT kegiatan kegiatan FROM mahasiswa ";
        ResultSet res = stat.executeQuery(sql);
        while(res.next()){
            cbKegiatan.addItem(res.getString("kegiatan"));
        }
    } catch(SQLException err){
        err.printStackTrace();
    }
}

private void refresh(){
    model = (DefaultTableModel) table.getModel();
    model.setRowCount(0);
    tCari.setText("");
    getDataTable();
    tJurusan.setText("");
}

private void reset(){
    tNim.setText("");
    tNama.setText("");
    tAlamat.setText("");
    tNotelp.setText("");
    tJurusan.setText("");
    tNim.setEditable(true);
    bSimpan.setEnabled(true);
    AutoNIM();
    AutoIDMhs();
}

private void insert(){
    PreparedStatement statement = null;
    String sql = "INSERT INTO mahasiswa (id_mahasiswa, nim, nama, alamat,
no_telp, jurusan, kegiatan)"
    + " VALUES(?, ?, ?, ?, ?, ?, ?)";
    try {
        statement = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, tID.getText());
        statement.setString(2, tNim.getText());
        statement.setString(3, tNama.getText());

```

```

        statement.setString(4, tAlamat.getText());
        statement.setString(5, tNotelp.getText());
        statement.setString(6, tJurusan.getText());
        statement.setString(7, cbKegiatan.getSelectedItem().toString());
        statement.executeUpdate();
    } catch (SQLException ex) {
        ex.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

```

```

private void update(){
    PreparedStatement statement = null;
    String sql = "UPDATE mahasiswa SET nim=?, nama=?, alamat=?, no_telp=?,
jurusan=?, kegiatan=? WHERE id_mahasiswa=?";
    try {
        statement = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, tNim.getText());
        statement.setString(2, tNama.getText());
        statement.setString(3, tAlamat.getText());
        statement.setString(4, tNotelp.getText());
        statement.setString(5, tJurusan.getText());
        statement.setString(6, cbKegiatan.getSelectedItem().toString());
        statement.setString(7, tID.getText());
        statement.executeUpdate();
    } catch (SQLException ex) {
        ex.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

```

```

private void delete(){
    PreparedStatement statement = null;
    String sql = "DELETE FROM mahasiswa WHERE id_mahasiswa=?";
    try {

```

```

        statement = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, tID.getText());
        statement.executeUpdate();
    } catch (SQLException ex) {
        ex.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

private void search(){
    model = (DefaultTableModel) table.getModel();
    PreparedStatement statement = null;
    try {
        String sql = "SELECT * FROM mahasiswa WHERE "
            + "id_mahasiswa AND nim like ?";
        statement = connection.prepareStatement(sql);
        statement.setString(1, "%" + tCari.getText() + "%");
        ResultSet res = statement.executeQuery();
        while(res.next()){
            Object[ ] obj = new Object[7];
            obj[0] = res.getString("id_mahasiswa");
            obj[1] = res.getString("nim");
            obj[2] = res.getString("nama");
            obj[3] = res.getString("alamat");
            obj[4] = res.getString("no_telp");
            obj[5] = res.getString("jurusan");
            obj[6] = res.getString("kegiatan");
            model.addRow(obj);
        }
    } catch(SQLException err){
        err.printStackTrace();
    }
}

private void bSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (!tID.getText().trim().isEmpty() &&
    !tNim.getText().trim().isEmpty() &&
    !tNama.getText().trim().isEmpty() &&
    !tAlamat.getText().trim().isEmpty() &&
    !tNotelp.getText().trim().isEmpty() &&
    !tJurusan.getText().trim().isEmpty() &&

```

```

!cbKegiatan.getSelectedItem().toString().trim().isEmpty() {
insert();
refresh();
reset();
JOptionPane.showMessageDialog(this,
    "Data Mahasiswa berhasil ditambahkan",
    "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
} else {
    JOptionPane.showMessageDialog(this,
        "Lengkapi Form terlebih dahulu!",
        "Notifikasi", JOptionPane.WARNING_MESSAGE);
}
}

private void bResetActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    reset();
}

private void bRefreshActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    refresh();
}

private void bUbahActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(!tNim.getText().trim().isEmpty()){
        if(!tID.getText().trim().isEmpty()&&
            !tNim.getText().trim().isEmpty()&&
            !tNama.getText().trim().isEmpty()&&
            !tAlamat.getText().trim().isEmpty()&&
            !tNotelp.getText().trim().isEmpty()&&
            !tJurusan.getText().trim().isEmpty()&&
            !cbKegiatan.getSelectedItem().toString().trim().isEmpty()){
            update();
            refresh();
            reset();
            JOptionPane.showMessageDialog(this,
                "Data Mahasiswa berhasil diubah",
                "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
        }else{
            JOptionPane.showMessageDialog(this,
                "Lengkapi form terlebih dahulu",
                "Notifikasi", JOptionPane.WARNING_MESSAGE);
        }
    }else{
        JOptionPane.showMessageDialog(this,
            "Pilih Data terlebih dahulu!",
```

```

        "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
}

private void bHapusActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(!tNim.getText().trim().isEmpty()) {
        int alert = JOptionPane.showConfirmDialog(this,
            "Anda yakin ingin menghapus SPP ini?",
            "Notifikasi", JOptionPane.YES_NO_OPTION,
            JOptionPane.QUESTION_MESSAGE);
        if(alert == JOptionPane.YES_OPTION) {
            delete();
            refresh();
            reset();
            JOptionPane.showMessageDialog(this,
                "Data SPP berhasil dihapus",
                "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(this,
            "Pilih data terlebih dahulu!",
            "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
}

private void bCariActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    model = (DefaultTableModel) table.getModel();
    model.setRowCount(0);
    search();
}

private void tableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    tID.setText(table.getModel().getValueAt(table.getSelectedRow(),0).toString());
    tNim.setText(table.getModel().getValueAt(table.getSelectedRow(),1).toString());
    tNama.setText(table.getModel().getValueAt(table.getSelectedRow(),2).toString());
    tAlamat.setText(table.getModel().getValueAt(table.getSelectedRow(),3).toString());

    tNotelp.setText(table.getModel().getValueAt(table.getSelectedRow(),4).toString());

    tJurusan.setText(table.getModel().getValueAt(table.getSelectedRow(),5).toString());

    cbKegiatan.setSelectedItem(table.getModel().getValueAt(table.getSelectedRow(),6)
        .toString());
    tNim.setEditable(false);
    tID.setEditable(false);
}

```

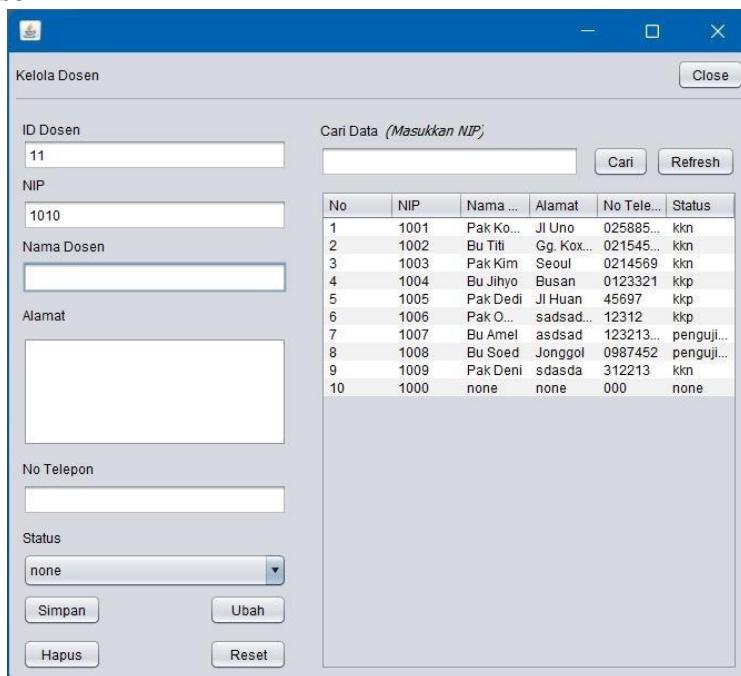
```

        bSimpan.setEnabled(false);
    }

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FPengguna p = new FPengguna();
    p.setVisible(true);
}

```

### 3.6 Frame dosen



Berikut source code frame dosen.

```

package view;
import koneksi.koneksi;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.table.DefaultTableModel;
import javax.swing.JOptionPane;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Fadhlhan
 */
public class FDosen extends javax.swing.JFrame {
    Connection connection;
    DefaultTableModel model;

```

```

/**
 * Creates new form FrameKelas
 */
public FDosen() {
    initComponents();
    connection = koneksi.getConnection();
    getDataTable();
//    getcbDosen();
    AutoID();
    tNama.requestFocus();
    generateAutoNumber();
    tNIP.setEditable(false);
    refresh();
}

private void getDataTable() {
    model = (DefaultTableModel) table.getModel();
    model.setRowCount(0);
    try {
        Statement stat = connection.createStatement();
        String sql = "SELECT * FROM dosen "
            + "WHERE id_dosen ";
        ResultSet res = stat.executeQuery(sql);
        while (res.next ()) {
            Object[ ] obj = new Object[6];
            obj[0] = res.getString("id_dosen");
            obj[1] = res.getString("nip");
            obj[2] = res.getString("nama_dosen");
            obj[3] = res.getString("alamat");
            obj[4] = res.getString("no_telp");
            obj[5] = res.getString("status");
            model.addRow(obj);
        }
    } catch (SQLException err) {
        err.printStackTrace();
    }
}

private void AutoID() {
    String sql = "SELECT MAX(id_dosen) AS max_id FROM dosen";
    try (Statement stat = connection.createStatement();
        ResultSet res = stat.executeQuery(sql)) {

        int newId;
        if (res.next()) {
            int maxId = res.getInt("max_id");
            newId = (maxId == 0) ? 1 : maxId + 1;
        } else {
            // Handle the case where no records exist yet
            newId = 1;
        }
    }
}

```

```

// Convert the ID to a string without leading zeros
String formattedId = Integer.toString(newId);
tID.setText(formattedId);

} catch (SQLException e) {
    e.printStackTrace();
}
}

private void generateAutoNumber() {
try {
    Statement stat = connection.createStatement();
    String sql = "SELECT MAX(nip) AS max_id FROM dosen";
    ResultSet res = stat.executeQuery(sql);

    if (res.next()) {
        int maxId = res.getInt("max_id");
        int newId = (maxId == 0) ? 1001 : maxId + 1;
        String formattedId = String.format("%04d", newId); // Format with leading zeros
        tNIP.setText(formattedId);
    } else {
        // Handle the case where no records exist yet
        tNIP.setText("1001");
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

private void refresh() {
model = (DefaultTableModel) table.getModel();
model.setRowCount(0);
getDataTable();
generateAutoNumber();
AutoID();
}

private void reset() {
tNIP.setText("");
tNama.setText("");
tNotelp.setText("");
tAlamat.setText("");
generateAutoNumber();
AutoID();
tNIP.setEditable(true);
bSimpan.setEnabled(true);
}

private void insert() {
PreparedStatement statement = null;

```

```

        String sql = "INSERT INTO dosen (id_dosen,nip,nama_dosen,alamat, no_telp, status)
" + "VALUES(?,?,?,?,?,?);";
        try {
            statement =
connection.prepareStatement(sql,Statement.RETURN_GENERATED_KEYS);
            statement.setString(1, tID.getText());
            statement.setString(2, tNIP.getText());
            statement.setString(3, tNama.getText());
            statement.setString(4, tAlamat.getText());
            statement.setString(5, tNotelp.getText());
            statement.setString(6, cbStatus.getSelectedItem().toString());
            statement.executeUpdate();
        } catch(SQLException ex) {
            ex.printStackTrace();
        } finally {
            try {
                statement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }

private void update() {
    PreparedStatement statement = null;
    String sql = "UPDATE dosen SET nip=? ,nama_dosen=? ,alamat=? ,no_telp=? ,status=?
" + "WHERE Id_dosen=?";
    try {
        statement =
connection.prepareStatement(sql,Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, tNIP.getText());
        statement.setString(2, tNama.getText());
        statement.setString(3, tAlamat.getText());
        statement.setString(4, tNotelp.getText());
        statement.setString(5, cbStatus.getSelectedItem().toString());
        statement.setString(6, tID.getText());
        statement.executeUpdate();
    } catch(SQLException ex) {
        ex.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

private void delete() {
    PreparedStatement statement = null;
    String sql = "DELETE FROM dosen WHERE id_dosen=?";

```

```

try {
    statement =
connection.prepareStatement(sql,Statement.RETURN_GENERATED_KEYS);
    statement.setString(1, tID.getText());
    statement.executeUpdate();
} catch(SQLException ex) {
    ex.printStackTrace();
} finally {
    try {
        statement.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}
}

private void search() {
model = (DefaultTableModel) table.getModel();
PreparedStatement statement = null;
try {
    String sql = "SELECT * FROM dosen WHERE id_dosen AND nip like ?";
    statement = connection.prepareStatement(sql);
    statement.setString(1, "%" + tCari.getText() + "%");
    ResultSet res = statement.executeQuery();
    while(res.next()) {
        Object[ ] obj = new Object[6];
        obj[0] = res.getString("id_dosen");
        obj[1] = res.getString("nip");
        obj[2] = res.getString("nama_dosen");
        obj[3] = res.getString("alamat");
        obj[4] = res.getString("no_telp");
        obj[5] = res.getString("status");
        model.addRow(obj);
    }
} catch(SQLException err) {
    err.printStackTrace();
}
}
}

```

**Di atas merupakan method yg akan digunakan pada button saat Action Performed :**

```

private void tNamaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void bSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(!tID.getText().trim().isEmpty() &&
    !tNIP.getText().trim().isEmpty() &&
    !tNama.getText().trim().isEmpty() &&
    !tAlamat.getText().trim().isEmpty() &&

```

```

        !tNotelp.getText().trim().isEmpty() &&
        !cbStatus.getSelectedItem().toString().trim().isEmpty()) {
        insert();
        refresh();
        reset();
        JOptionPane.showMessageDialog(this,
            "Data Dosen berhasil ditambahkan",
            "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
    } else {
        JOptionPane.showMessageDialog(this,
            "Lengkapi form terlebih dahulu!",
            "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
}

private void bUbahActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(!tNIP.getText().trim().isEmpty()) {
        if(!tID.getText().trim().isEmpty() &&
           !tNIP.getText().trim().isEmpty() &&
           !tNama.getText().trim().isEmpty() &&
           !tAlamat.getText().trim().isEmpty() &&
           !tNotelp.getText().trim().isEmpty() &&
           !cbStatus.getSelectedItem().toString().trim().isEmpty()) {
            update();
            refresh();
            reset();
            JOptionPane.showMessageDialog(this,
                "Data Dosen berhasil diubah",
                "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this,
                "Lengkapi form terlebih dahulu!",
                "Notifikasi", JOptionPane.WARNING_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(this,
            "Pilih data terlebih dahulu!",
            "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
}

private void bHapusActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(!tNIP.getText().trim().isEmpty()) {
        int alert = JOptionPane.showConfirmDialog(this,
            "Anda yakin ingin menghapus Dosen ini?",
            "Notifikasi", JOptionPane.YES_NO_OPTION,
            JOptionPane.QUESTION_MESSAGE);
        if(alert == JOptionPane.YES_OPTION) {
            delete();
        }
    }
}

```

```

refresh();
reset();
JOptionPane.showMessageDialog(this,
    "Data Dosen berhasil dihapus",
    "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
}
} else {
JOptionPane.showMessageDialog(this,
    "Pilih data terlebih dahulu!",
    "Notifikasi", JOptionPane.WARNING_MESSAGE);
}
}

private void bResetActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
reset();
}

private void bRefreshActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
refresh();
}

private void tableMouseClicked(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
tID.setText(table.getModel().getValueAt(table.getSelectedRow(), 0).toString());
tNIP.setText(table.getModel().getValueAt(table.getSelectedRow(), 1).toString());
tNama.setText(table.getModel().getValueAt(table.getSelectedRow(), 2).toString());
tAlamat.setText(table.getModel().getValueAt(table.getSelectedRow(), 3).toString());
tNotelp.setText(table.getModel().getValueAt(table.getSelectedRow(), 4).toString());

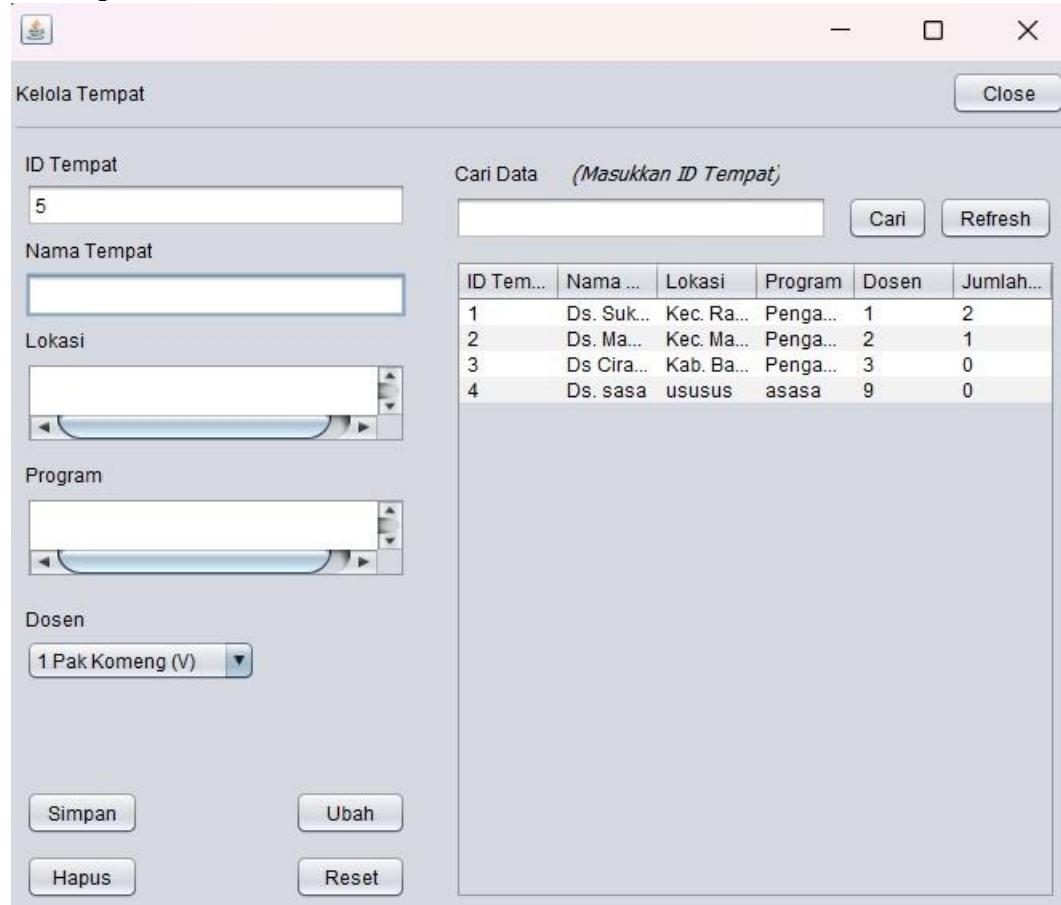
cbStatus.setSelectedItem(table.getModel().getValueAt(table.getSelectedRow(),5).toString());
tNIP.setEditable(false);
tID.setEditable(false);
bSimpan.setEnabled(false);
}

private void bCariActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
model = (DefaultTableModel) table.getModel();
model.setRowCount(0);
search();
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
dispose();
}

```

### 3.7 Frame Tempat kkn



```
package view;  
import koneksi.koneksi;  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import javax.swing.table.DefaultTableModel;  
import javax.swing.JOptionPane;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import java.util.HashSet;  
import java.util.Set;  
  
/**  
 *  
 * @author Fadhlhan
```

```

        */

public class FTempatkkn extends javax.swing.JFrame {
    Connection connection;
    DefaultTableModel model;

    /**
     * Creates new form FrameKelas
     */
    public FTempatkkn() {
        initComponents();
        connection = koneksi.getConnection();
        tNamatempat.requestFocus();
        getcbDosen();
        getDataTable();
        generateAutoNumber();
        tID.setEditable(false);
    }

    private void getDataTable() {
        model = (DefaultTableModel) table.getModel();
        model.setRowCount(0);
        try {
            Statement stat = connection.createStatement();
            String sql = "SELECT * FROM tempat_kkn, dosen "
                    + "WHERE tempat_kkn.id_dosen=dosen.id_dosen ";
            ResultSet res = stat.executeQuery(sql);
            while (res.next()) {
                Object[] obj = new Object[6];
                obj[0] = res.getString("id_tempat");
                obj[1] = res.getString("nama_tempat");
                obj[2] = res.getString("lokasi");
                obj[3] = res.getString("program");
                obj[4] = res.getString("id_dosen");
                obj[5] = res.getString("jumlah_mahasiswa");
                model.addRow(obj);
            }
        }
    }
}

```

```

        }

    } catch (SQLException err) {
        err.printStackTrace();
    }
}

private void generateAutoNumber() {

    try {
        Statement stat = connection.createStatement();

        String sql = "SELECT MAX(id_tempat) AS max_id FROM tempat_kkn";
        ResultSet res = stat.executeQuery(sql);

        if (res.next()) {
            int maxId = res.getInt("max_id");
            int newId = maxId + 1;
            tID.setText(String.valueOf(newId));
        } else {
            // Handle the case where no records exist yet
            tID.setText("1");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void refresh() {
    model = (DefaultTableModel) table.getModel();
    model.setRowCount(0);
    getcbDosen();
    getDataTable();
    generateAutoNumber();
}

private void reset() {
}

```

```

tNamatempat.setText("");
tProgram.setText("");
tID.setText("");
tLokasi.setText("");
getcbDosen();
tNamatempat.setEditable(true);
bSimpan.setEnabled(true);
generateAutoNumber();
}

private void insert() {
    PreparedStatement statement = null;
    String sql = "INSERT INTO tempat_kkn (id_tempat,
nama_tempat,lokasi,program,id_dosen,jumlah_mahasiswa) " +
"VALUES(?,?,?,?,?,0);";
    try {
        statement =
connection.prepareStatement(sql,Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, tID.getText());
        statement.setString(2, tNamatempat.getText());
        statement.setString(3, tLokasi.getText());
        statement.setString(4, tProgram.getText());
        statement.setInt(5, getIDDosen(cbDosen.getSelectedItem().toString()));
        statement.executeUpdate();
    } catch(SQLException ex) {
        ex.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

private int getIDDosen(String selectedItem) {

```

```

int id = 0;
if (selectedItem != null && selectedItem.contains(" ")) {
    // Memisahkan ID dan nama dari format "ID_NAMA"
    String[] parts = selectedItem.split(" ", 2);
    try {
        id = Integer.parseInt(parts[0]); // ID adalah bagian pertama
    } catch (NumberFormatException e) {
        e.printStackTrace();
    }
}
return id;
}

```

```

private int extractIDFromComboBox(String item) {
    // Memisahkan ID dan nama dari format "ID_NAMA"
    String[] parts = item.split(" ", 2);
    try {
        return Integer.parseInt(parts[0]); // ID adalah bagian pertama
    } catch (NumberFormatException e) {
        e.printStackTrace();
    }
    return 0; // Return 0 atau nilai default jika parsing gagal
}

```

```

private void getcbDosen() {
    cbDosen.removeAllItems(); // Hapus semua item dari combo box

    // Ambil ID dosen yang sudah terdaftar dari tabel tempat_kkn
    Set<Integer> registeredDosenIDs = new HashSet<>();
    String checkRegisteredSQL = "SELECT id_dosen FROM tempat_kkn";

    try (Statement stat = connection.createStatement()) {
        ResultSet res = stat.executeQuery(checkRegisteredSQL)) {

```

```

        while (res.next()) {
            registeredDosenIDs.add(res.getInt("id_dosen"));
        }

    } catch (SQLException err) {
        Logger.getLogger(getClass().getName()).log(Level.SEVERE, "SQL Error in
getcbDosen", err);
    }

    // Ambil semua dosen dari tabel dosen
    String sql = "SELECT id_dosen, nama_dosen FROM dosen WHERE status =
'kkn';

    try (Statement stat = connection.createStatement());
        ResultSet res = stat.executeQuery(sql)) {

            while (res.next()) {
                int idDosen = res.getInt("id_dosen");
                String namaDosen = res.getString("nama_dosen");
                if (registeredDosenIDs.contains(idDosen)) {
                    cbDosen.addItem(idDosen + " " + namaDosen + " (V)"); // Dosen yang
sudah terdaftar
                } else {
                    cbDosen.addItem(idDosen + " " + namaDosen); // Dosen yang belum
terdaftar
                }
            }

        } catch (SQLException err) {
            Logger.getLogger(getClass().getName()).log(Level.SEVERE, "SQL Error in
getcbDosen", err);
        }
    }
}

```

```

private void update() {
    String sql = "UPDATE tempat_kkn SET nama_tempat = ?, lokasi = ?, program = ?, id_dosen = ? WHERE id_tempat = ?";

    try (PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setString(1, tNamatempat.getText().trim());
        statement.setString(2, tLokasi.getText().trim());
        statement.setString(3, tProgram.getText().trim());

        // Mengambil item yang dipilih dan mengekstrak ID dosen
        String selectedDosen = cbDosen.getSelectedItem().toString();
        int idDosen = extractIDFromComboBox(selectedDosen);

        System.out.println("Updating with ID Dosen: " + idDosen); // Baris debugging

        statement.setInt(4, idDosen);
        statement.setString(5, tID.getText().trim());

        int rowsAffected = statement.executeUpdate();
        System.out.println("Rows affected: " + rowsAffected); // Baris debugging
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

private void delete() {
    PreparedStatement statement = null;
    String sql = "DELETE FROM tempat_kkn WHERE id_tempat=?";
    try {
        statement =
connection.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, tID.getText());
    }

```

```

        statement.executeUpdate();
    } catch(SQLException ex) {
        ex.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

private void search() {
    model = (DefaultTableModel) table.getModel();
    PreparedStatement statement = null;
    try {
        String sql = "SELECT * FROM dosen,tempat_kkn "
                + "WHERE tempat_kkn.id_dosen=dosen.id_dosen AND id_tempat like ?";
        statement = connection.prepareStatement(sql);
        statement.setString(1, "%" + tCari.getText() + "%");
        ResultSet res = statement.executeQuery();
        while(res.next()) {
            Object[ ] obj = new Object[4];
            obj[0] = res.getString("id_tempat");
            obj[1] = res.getString("nama_tempat");
            obj[2] = res.getString("lokasi");
            obj[3] = res.getString("nama_dosen");
            model.addRow(obj);
        }
    } catch(SQLException err) {
        err.printStackTrace();
    }
}

private void bSimpanActionPerformed(java.awt.event.ActionEvent evt) {

```

```

// TODO add your handling code here:

if (!tNamatempat.getText().trim().isEmpty() &&
    !tLokasi.getText().trim().isEmpty() &&
    !tProgram.getText().trim().isEmpty() &&
    cbDosen.getSelectedItem() != null) {

    // Ambil dosen yang dipilih
    String selectedDosen = (String) cbDosen.getSelectedItem();
    if (selectedDosen != null && selectedDosen.contains("(V)")) {
        JOptionPane.showMessageDialog(this,
            "Dosen yang dipilih sudah terdaftar. Silakan pilih dosen lain.",
            "Peringatan", JOptionPane.WARNING_MESSAGE);
        return; // Keluar dari metode untuk mencegah pengiriman
    }

    // Ambil ID dosen dari item yang dipilih di cbDosen
    int idDosen = getIDDosen(selectedDosen);

    // Lakukan penyimpanan dengan idDosen dan data lainnya
    insert();
    refresh();
    reset();
    JOptionPane.showMessageDialog(this,
        "Data Tempat berhasil ditambahkan",
        "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
} else {
    JOptionPane.showMessageDialog(this,
        "Lengkapi form terlebih dahulu!",
        "Notifikasi", JOptionPane.WARNING_MESSAGE);
}
}

private void bUbahActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

```

```

        if (!tID.getText().trim().isEmpty()) {
            if (!tNamatempat.getText().trim().isEmpty() &&
                !tLokasi.getText().trim().isEmpty() &&
                !tProgram.getText().trim().isEmpty() &&
                cbDosen.getSelectedItem() != null) {

                update(); // Ensure update method works as expected

                // Refresh and reset UI
                refresh();
                reset();

                // Show success message
                JOptionPane.showMessageDialog(this,
                    "Data Tempat berhasil diubah",
                    "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this,
                    "Lengkapi form terlebih dahulu!",
                    "Notifikasi", JOptionPane.WARNING_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(this,
                "Pilih data terlebih dahulu!",
                "Notifikasi", JOptionPane.WARNING_MESSAGE);
        }
    }

private void bHapusActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(!tID.getText().trim().isEmpty()) {
        int alert = JOptionPane.showConfirmDialog(this,
            "Anda yakin ingin menghapus Data ini?",
            "Notifikasi", JOptionPane.YES_NO_OPTION,
            JOptionPane.QUESTION_MESSAGE);
    }
}

```

```

        if(alert == JOptionPane.YES_OPTION) {
            delete();
            refresh();
            reset();
            JOptionPane.showMessageDialog(this,
                "Data Tempat berhasil dihapus",
                "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(this,
            "Pilih data terlebih dahulu!",
            "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
}

private void bResetActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    reset();
}

private void bRefreshActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    refresh();
}

private void tableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    String idTempat = table.getModel().getValueAt(table.getSelectedRow(),
0).toString();
    String namaTempat = table.getModel().getValueAt(table.getSelectedRow(),
1).toString();
    String lokasi = table.getModel().getValueAt(table.getSelectedRow(), 2).toString();
    String program = table.getModel().getValueAt(table.getSelectedRow(),
3).toString();
    String namaDosen = table.getModel().getValueAt(table.getSelectedRow(),
4).toString();
}

```

```

// Set data ke text fields
tID.setText(idTempat);
tNamatempat.setText(namaTempat);
tLokasi.setText(lokasi);
tProgram.setText(program);

// Cek item yang ada di combo box
for (int i = 0; i < cbDosen.getItemCount(); i++) {
    String item = cbDosen.getItemAt(i).toString();
    // Asumsi item di combo box dalam format "ID_NAMA"
    if (item.contains(namaDosen)) {
        cbDosen.setSelectedIndex(i);
        break;
    }
}

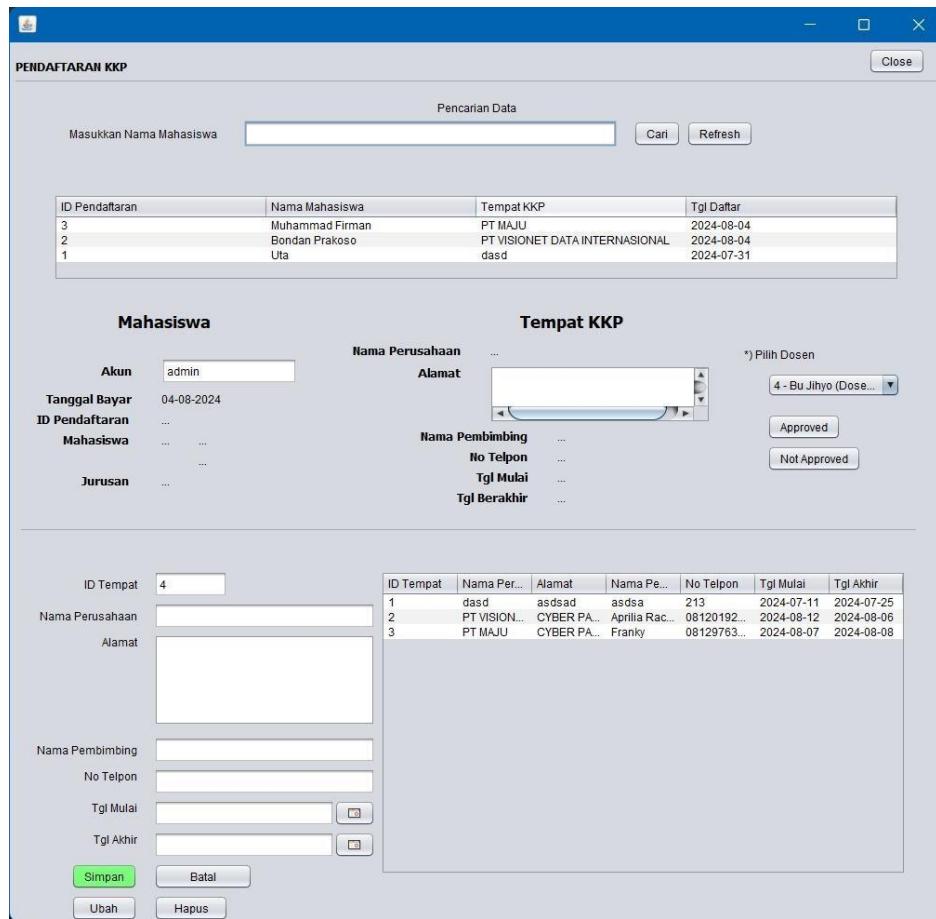
tID.setEditable(false);
bSimpan.setEnabled(false);
}

private void bCariActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    model = (DefaultTableModel) table.getModel();
    model.setRowCount(0);
    search();
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    dispose();
}

```

### 3.8 Frame tempat kkp



**Berikut source code Frame Tempat KKP yg akan digunakan oleh admin ketika tempat kkp telah di setujui :**

```
package view;
import koneksi.koneksi;
import session.session;
import java.sql.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

```

/**
 *
 * @author fadhl
 */
public class FTempatkkp extends javax.swing.JFrame {
    Connection connection;
    DefaultTableModel model,model2,model3;
    /**
     * Creates new form FDaftarkkp
     */
    public FTempatkkp() {
        initComponents();
        connection = koneksi.getConnection();
        setTanggal();
        tabelPendaftaran();
        tabeltempatkkp();
        tCari.requestFocus();
        generateAutoNumber();
        // refresh();
        akunUser();
        getcbDosen();
    }

    public void setTanggal() {
        // Mendapatkan tanggal saat ini
        LocalDate today = LocalDate.now();

        // Menentukan format tanggal
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");

        // Mengonversi tanggal ke format yang diinginkan
        String formattedDate = today.format(formatter);

        // Menetapkan tanggal yang diformat ke komponen
        ITanggal.setText(formattedDate);
    }

    private void tabelPendaftaran() {
        model = (DefaultTableModel) table.getModel();
        model.setRowCount(0); // Menghapus data lama

        // SQL query untuk mengambil data dari tabel pendaftaran_kkp dan tabel lainnya
        // dengan alias yang sesuai
        String sql = "SELECT pendaftaran_kkp.id_pendaftaran, mahasiswa.nama,
tempat_kkp.nama_perusahaan, pendaftaran_kkp.tgl_daftar "
        + "FROM pendaftaran_kkp "
        + "JOIN mahasiswa ON pendaftaran_kkp.id_mahasiswa =
mahasiswa.id_mahasiswa "
    }
}

```

```

+ "JOIN tempat_kkp ON pendaftaran_kkp.id_tempat = tempat_kkp.id_tempat "
+ "ORDER BY pendaftaran_kkp.id_pendaftaran DESC";

try (Statement stat = connection.createStatement());
    ResultSet res = stat.executeQuery(sql)) {

    while (res.next()) {
        // Ambil data dari ResultSet
        Object[] obj = new Object[4];
        obj[0] = res.getString("id_pendaftaran"); // ID Pendaftaran
        obj[1] = res.getString("nama");          // Nama Mahasiswa
        obj[2] = res.getString("nama_perusahaan"); // Nama Tempat
        obj[3] = res.getDate("tgl_daftar");      // Tanggal Daftar (gunakan getDate untuk
tipe SQL Date)

        // Tambahkan baris ke model tabel
        model.addRow(obj);
    }
} catch (SQLException err) {
    err.printStackTrace();
    JOptionPane.showMessageDialog(null,
        "Terjadi kesalahan saat memuat data pendaftaran.",
        "Kesalahan",
        JOptionPane.ERROR_MESSAGE);
}
}

private void akunUser() {
    // Ambil username pengguna dari SessionManager
    String username = session.getInstance().getUsername();
    // Update teks label dengan username
    lblUsername.setText(username);
}

private void search() {
    model = (DefaultTableModel) table.getModel();
    model.setRowCount(0); // Menghapus data lama

    // SQL query dengan kondisi pencarian berdasarkan nama mahasiswa
    String sql = "SELECT pendaftaran_kkp.id_pendaftaran, mahasiswa.nama,
tempat_kkp.nama_perusahaan AS nama_tempat, pendaftaran_kkp.tgl_daftar "
        + "FROM pendaftaran_kkp "
        + "JOIN mahasiswa ON pendaftaran_kkp.id_mahasiswa =
mahasiswa.id_mahasiswa "
        + "JOIN tempat_kkp ON pendaftaran_kkp.id_tempat = tempat_kkp.id_tempat
"

```

```

+ "WHERE mahasiswa.nama LIKE ?";

try (PreparedStatement pst = connection.prepareStatement(sql)) {
    // Menetapkan parameter untuk pencarian
    String searchKeyword = "%" + tCari.getText().trim() + "%";
    pst.setString(1, searchKeyword);

    // Menjalankan query dan memproses hasil
    try (ResultSet rs = pst.executeQuery()) {
        while (rs.next()) {
            Object[] obj = new Object[4];
            obj[0] = rs.getString("id_pendaftaran");
            obj[1] = rs.getString("nama");
            obj[2] = rs.getString("nama_tempat");
            obj[3] = rs.getString("tgl_daftar");
            model.addRow(obj);
        }
    }
} catch (SQLException e) {
    e.printStackTrace();
    JOptionPane.showMessageDialog(this,
        "Terjadi kesalahan saat melakukan pencarian.",
        "Error", JOptionPane.ERROR_MESSAGE);
}
}

private void refresh() {
    model = (DefaultTableModel) table.getModel();
    model.setRowCount(0);
    model2 = (DefaultTableModel) table2.getModel();
    model2.setRowCount(0);
    tCari.setText("");
    tabelPendaftaran();
    tabeltempatkkp();
    getcbDosen();
}

private void tabeltempatkkp() {
    model2 = (DefaultTableModel) table2.getModel();
    model2.setRowCount(0);
    try {
        Statement stat = connection.createStatement();
        String sql = "SELECT * FROM tempat_kkp WHERE id_tempat ";

        ResultSet res = stat.executeQuery(sql);
        while (res.next()) {
            Object[] obj = new Object[7];

```

```

        obj[0] = res.getString("id_tempat");
        obj[1] = res.getString("nama_perusahaan");
        obj[2] = res.getString("alamat");
        obj[3] = res.getString("nama_pembimbing");
        obj[4] = res.getString("notelp");
        obj[5] = res.getString("tgl_mulai");
        obj[6] = res.getString("tgl_berakhir");
        model2.addRow(obj);
    }
} catch (SQLException err) {
    err.printStackTrace();
}
}

private void getcbDosen() {
    cbDosen.removeAllItems(); // Kosongkan JComboBox sebelum menambah item
    String sqlSelectDosen = "SELECT id_dosen, nama_dosen FROM dosen WHERE
status = 'kkp'";
    String sqlCheckDosenAssigned = "SELECT COUNT(*) FROM pendaftaran_kkpdetail
WHERE id_dosen = ?";

    try (Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(sqlSelectDosen)) {

        while (resultSet.next()) {
            String idDosen = resultSet.getString("id_dosen");
            String namaDosen = resultSet.getString("nama_dosen");

            // Periksa apakah dosen sudah terdaftar di pendaftaran_kkpdetail
            try (PreparedStatement checkStatement =
connection.prepareStatement(sqlCheckDosenAssigned)) {
                checkStatement.setString(1, idDosen);
                try (ResultSet checkResultSet = checkStatement.executeQuery()) {
                    if (checkResultSet.next() && checkResultSet.getInt(1) > 0) {
                        namaDosen += "";
                    }
                }
            }
        }

        // Tambah item ke JComboBox
        cbDosen.addItem(idDosen + " - " + namaDosen);
    }
} catch (SQLException e) {
    e.printStackTrace();
    JOptionPane.showMessageDialog(null,
    "Terjadi kesalahan saat memuat data dosen.",
    "Kesalahan",

```

```

        JOptionPane.ERROR_MESSAGE);
    }
}

private void generateAutoNumber() {
    try {
        Statement stat = connection.createStatement();
        String sql = "SELECT MAX(id_tempat) AS max_id FROM tempat_kkp";
        ResultSet res = stat.executeQuery(sql);

        if (res.next()) {
            int maxId = res.getInt("max_id");
            int newId = maxId + 1;
            tID.setText(String.valueOf(newId));
        } else {
            // Handle the case where no records exist yet
            tID.setText("1");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void insertTempatKkp() {
    PreparedStatement statement = null;
    ResultSet resultSet = null;

    // SQL untuk memasukkan data ke tabel tempat_kkp
    String sqlInsert = "INSERT INTO tempat_kkp "
        + "(id_tempat, nama_perusahaan, alamat, nama_pembimbing, notelp,
    tgl_mulai, tgl_berakhir)"
        + "VALUES (?, ?, ?, ?, ?, ?, ?);"

    // SQL untuk memeriksa apakah ID Tempat sudah ada
    String sqlCheck = "SELECT COUNT(*) FROM tempat_kkp WHERE id_tempat =
?";

    // Ambil data dari GUI
    String idTempat = tID.getText().trim();
    String namaPerusahaan = tNmPerusahaan.getText().trim();
    String alamat = tAlamat.getText().trim();
    String namaPembimbing = tNmPembimbing.getText().trim();
    String notelp = tNotelp.getText().trim();
    java.sql.Date tglMulai = new java.sql.Date(jdcTglMulai.getDate().getTime());
    java.sql.Date tglBerakhir = new java.sql.Date(jdcTglakhir.getDate().getTime());

    // Validasi ID Tempat tidak kosong

```

```

if (idTempat.isEmpty()) {
    JOptionPane.showMessageDialog(null,
        "ID Tempat tidak boleh kosong.",
        "Kesalahan Validasi",
        JOptionPane.ERROR_MESSAGE);
    return;
}

try {
    // Cek apakah ID Tempat sudah ada
    try (PreparedStatement checkStatement =
connection.prepareStatement(sqlCheck)) {
        checkStatement.setString(1, idTempat);
        try (ResultSet checkResultSet = checkStatement.executeQuery()) {
            if (checkResultSet.next() && checkResultSet.getInt(1) > 0) {
                // Jika ID Tempat sudah ada
                JOptionPane.showMessageDialog(null,
                    "ID Tempat sudah ada.",
                    "Informasi",
                    JOptionPane.INFORMATION_MESSAGE);
                return;
            }
        }
    }
}

// Jika ID Tempat belum ada, lanjutkan dengan penyisipan data
try (PreparedStatement insertStatement = connection.prepareStatement(sqlInsert))
{
    insertStatement.setString(1, idTempat); // ID Tempat
    insertStatement.setString(2, namaPerusahaan); // Nama Perusahaan
    insertStatement.setString(3, alamat); // Alamat
    insertStatement.setString(4, namaPembimbing); // Nama Pembimbing
    insertStatement.setString(5, notelp); // No Telp
    insertStatement.setDate(6, tglMulai); // Tanggal Mulai
    insertStatement.setDate(7, tglBerakhir); // Tanggal Berakhir
    insertStatement.executeUpdate();

    // Notifikasi bahwa data berhasil disimpan
    JOptionPane.showMessageDialog(null,
        "Data berhasil disimpan.",
        "Informasi",
        JOptionPane.INFORMATION_MESSAGE);
    System.out.println("Insert successful");

    // Aktifkan tombol bDaftar jika data berhasil disimpan
    bUbah.setEnabled(true);
    bHapus.setEnabled(true);
}

```

```

        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null,
                "Terjadi kesalahan saat menyimpan data.",
                "Kesalahan",
                JOptionPane.ERROR_MESSAGE);

    } finally {
        try {
            if (resultSet != null) {
                resultSet.close();
            }
            if (statement != null) {
                statement.close();
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

private void reset() {
    tID.setText("");
    tNmPerusahaan.setText("");
    tAlamat.setText("");
    tNmPembimbing.setText("");
    tNotelp.setText("");
    generateAutoNumber();
    bSimpan.setEnabled(true);
}

private void updateTempatKkp() {
// Mengambil data dari text fields
    String idTemp = tID.getText().trim();
    String namaPerusahaan = tNmPerusahaan.getText().trim();
    String alamat = tAlamat.getText().trim();
    String namaPembimbing = tNmPembimbing.getText().trim();
    String notelp = tNotelp.getText().trim();
    java.util.Date tglMulai = jdcTglMulai.getDate();
    java.util.Date tglAkhir = jdcTglakhir.getDate();

// Validasi data (misalnya, periksa apakah ID Tempat tidak kosong)
    if (idTemp.isEmpty()) {
        JOptionPane.showMessageDialog(this, "ID Tempat tidak boleh kosong!",
            "Peringatan", JOptionPane.WARNING_MESSAGE);
        return;
    }
}

```

```

// Query SQL untuk update data pada tabel tempat_kkp
String sql = "UPDATE tempat_kkp SET "
        + "nama_perusahaan = ?, "
        + "alamat = ?, "
        + "nama_pembimbing = ?, "
        + "notelp = ?, "
        + "tgl_mulai = ?, "
        + "tgl_berakhir = ? "
        + "WHERE id_tempat = ?";

try (PreparedStatement pst = connection.prepareStatement(sql)) {
    // Menetapkan parameter untuk PreparedStatement
    pst.setString(1, namaPerusahaan);
    pst.setString(2, alamat);
    pst.setString(3, namaPembimbing);
    pst.setString(4, notelp);
    pst.setDate(5, new java.sql.Date(tglMulai.getTime())); // Mengonversi
    java.util.Date ke java.sql.Date
    pst.setDate(6, new java.sql.Date(tglAkhir.getTime())); // Mengonversi
    java.util.Date ke java.sql.Date
    pst.setString(7, idTemp);

    // Menjalankan update query
    int rowsAffected = pst.executeUpdate();
    if (rowsAffected > 0) {
        JOptionPane.showMessageDialog(this, "Data berhasil diperbarui!",
        "Informasi", JOptionPane.INFORMATION_MESSAGE);
    } else {
        JOptionPane.showMessageDialog(this, "Gagal memperbarui data. ID Tempat
tidak ditemukan.", "Error", JOptionPane.ERROR_MESSAGE);
    }
} catch (SQLException e) {
    // Menangani kesalahan SQL
    e.printStackTrace();
    JOptionPane.showMessageDialog(this, "Terjadi kesalahan saat memperbarui
data.", "Error", JOptionPane.ERROR_MESSAGE);
}
}

private void deleteTempatKkp() {
    // Mengambil ID Tempat dari text field
    String idTemp = tID.getText().trim();

    // Validasi data (misalnya, periksa apakah ID Tempat tidak kosong)
    if (idTemp.isEmpty()) {
        JOptionPane.showMessageDialog(this, "ID Tempat tidak boleh kosong!",
        "Peringatan", JOptionPane.WARNING_MESSAGE);
    }
}

```

```

        return;
    }

    // Menampilkan dialog konfirmasi
    int confirm = JOptionPane.showConfirmDialog(
        this,
        "Apakah Anda yakin ingin menghapus data dengan ID Tempat: " + idTemp + "?",
        "Konfirmasi Penghapusan",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE
    );

    // Jika pengguna memilih "YES", lanjutkan dengan penghapusan
    if (confirm == JOptionPane.YES_OPTION) {
        // Query SQL untuk menghapus data dari tabel tempat_kkp
        String sql = "DELETE FROM tempat_kkp WHERE id_tempat = ?";

        try (PreparedStatement pst = connection.prepareStatement(sql)) {
            // Menetapkan parameter untuk PreparedStatement
            pst.setString(1, idTemp);

            // Menjalankan delete query
            int rowsAffected = pst.executeUpdate();
            if (rowsAffected > 0) {
                JOptionPane.showMessageDialog(this, "Data berhasil dihapus!",
                    "Informasi", JOptionPane.INFORMATION_MESSAGE);
                // Mengosongkan text field setelah penghapusan
                tID.setText("");
                tNmPerusahaan.setText("");
                tAlamat.setText("");
                tNmPembimbing.setText("");
                tNotelp.setText("");
                jdcTglMulai.setDate(null);
                jdcTglakhir.setDate(null);
            } else {
                JOptionPane.showMessageDialog(this, "Gagal menghapus data. ID Tempat
tidak ditemukan.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (SQLException e) {
            // Menangani kesalahan SQL
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Terjadi kesalahan saat menghapus
data.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    } else {
        // Pengguna memilih "NO", tidak melakukan penghapusan
        JOptionPane.showMessageDialog(this, "Penghapusan data dibatalkan.",
            "Informasi", JOptionPane.INFORMATION_MESSAGE);
    }
}

```

```

        }
    }

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    dispose();
}

private void bCariMhsActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (!tCari.getText().trim().isEmpty()) {
        search();
    } else {
        JOptionPane.showMessageDialog(this,
            "Masukkan NIM yang ingin dicari!",
            "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
}

private void bRefreshActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    refresh();
}

private void tableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    int row = table.getSelectedRow();
    String idPendaftaran = table.getValueAt(row, 0).toString();

    // Query SQL untuk mendapatkan detail berdasarkan ID Pendaftaran
    String sql = "SELECT pendaftaran_kkp.id_pendaftaran,
pendaftaran_kkp.id_mahasiswa, mahasiswa.nim, mahasiswa.nama AS nama_mahasiswa,
mahasiswa.jurusan,"
        + "tempat_kkp.nama_perusahaan, tempat_kkp.alamat,
tempat_kkp.nama_pembimbing, tempat_kkp.notelp, "
        + "tempat_kkp.tgl_mulai, tempat_kkp.tgl_berakhir"
        + "FROM pendaftaran_kkp "
        + "JOIN mahasiswa ON pendaftaran_kkp.id_mahasiswa =
mahasiswa.id_mahasiswa"
        + "JOIN tempat_kkp ON pendaftaran_kkp.id_tempat = tempat_kkp.id_tempat"
        + "WHERE pendaftaran_kkp.id_pendaftaran = ?";

    try (PreparedStatement pst = connection.prepareStatement(sql)) {
        // Menetapkan parameter untuk PreparedStatement
        pst.setString(1, idPendaftaran);

        // Menjalankan query dan memproses hasil
        try (ResultSet rs = pst.executeQuery()) {

```

```

        if (rs.next()) {
            // Memperbarui label dengan data yang diperoleh
            LidPendaftaran.setText(rs.getString("id_pendaftaran"));
            LidMahasiswa.setText(rs.getString("id_mahasiswa"));
            Lnim.setText(rs.getString("nim"));
            LnamaMahasiswa.setText(rs.getString("nama_mahasiswa"));
            Ljurusan.setText(rs.getString("jurusan"));
            LnmPerusahaan.setText(rs.getString("nama_perusahaan"));
            Lalamat.setText(rs.getString("alamat"));
            LnmPembimbing.setText(rs.getString("nama_pembimbing"));
            Lnotelp.setText(rs.getString("notelp"));
            LtglMulai.setText(rs.getDate("tgl_mulai").toString());
            LtglAkhir.setText(rs.getDate("tgl_berakhir").toString());
        } else {
            // Menangani kasus jika data tidak ditemukan
            JOptionPane.showMessageDialog(this,
                "Data tidak ditemukan!",
                "Notifikasi", JOptionPane.INFORMATION_MESSAGE);

            // Mengosongkan label jika data tidak ditemukan
            LidPendaftaran.setText("");
            LidMahasiswa.setText("");
            Lnim.setText("");
            LnamaMahasiswa.setText("");
            Ljurusan.setText("");
            LnmPerusahaan.setText("");
            Lalamat.setText("");
            LnmPembimbing.setText("");
            Lnotelp.setText("");
            LtglMulai.setText("");
            LtglAkhir.setText("");
        }
    }
} catch (SQLException e) {
    // Menangani kesalahan SQL
    e.printStackTrace();
    JOptionPane.showMessageDialog(this,
        "Terjadi kesalahan saat mengambil data.",
        "Error", JOptionPane.ERROR_MESSAGE);
}
}

private void bNotApproveActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String idPendaftaran = LidPendaftaran.getText().trim(); // Contoh: ID Pendaftaran
    diambil dari text field

    if (idPendaftaran.isEmpty()) {

```

```

JOptionPane.showMessageDialog(null,
    "ID Pendaftaran tidak boleh kosong.",
    "Kesalahan Validasi",
    JOptionPane.ERROR_MESSAGE);
return;
}

// Query SQL untuk menghapus data dari tabel pendaftaran_kkp
String sqlDelete = "DELETE FROM pendaftaran_kkp WHERE id_pendaftaran = ?";

try (PreparedStatement deleteStatement = connection.prepareStatement(sqlDelete)) {
    // Set parameter untuk query
    deleteStatement.setString(1, idPendaftaran);

    // Jalankan query dan ambil jumlah baris yang dipengaruhi
    int rowsAffected = deleteStatement.executeUpdate();

    if (rowsAffected > 0) {
        // Notifikasi bahwa data berhasil dihapus
        JOptionPane.showMessageDialog(null,
            "Data berhasil dihapus.",
            "Informasi",
            JOptionPane.INFORMATION_MESSAGE);
    } else {
        // Tidak ada baris yang dihapus
        JOptionPane.showMessageDialog(null,
            "ID Pendaftaran tidak ditemukan.",
            "Informasi",
            JOptionPane.INFORMATION_MESSAGE);
    }
    refresh();
} catch (SQLException e) {
    // Tampilkan pesan kesalahan jika terjadi SQLException
    e.printStackTrace();
    JOptionPane.showMessageDialog(null,
        "Terjadi kesalahan saat menghapus data.",
        "Kesalahan",
        JOptionPane.ERROR_MESSAGE);
}
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    String selectedDosen = (String) cbDosen.getSelectedItem();

    if (selectedDosen != null) {
        // Ambil ID Dosen dari teks cbDosen
        String[] parts = selectedDosen.split(" - ");
        String idDosen = parts[0]; // ID Dosen adalah bagian pertama dari string
    }
}

```

```

// Ambil ID Pendaftaran dari GUI atau sumber lain yang sesuai
String idPendaftaran = LidPendaftaran.getText().trim(); // Contoh: ID Pendaftaran
diambil dari text field

if (idPendaftaran.isEmpty()) {
    JOptionPane.showMessageDialog(null,
        "ID Pendaftaran tidak boleh kosong.",
        "Kesalahan Validasi",
        JOptionPane.ERROR_MESSAGE);
    return;
}

// Update tabel pendaftaran_kkpdetail dengan ID Dosen dan status 'approve'
String sqlUpdate = "UPDATE pendaftaran_kkpdetail SET id_dosen = ?, status =
'approve' WHERE id_pendaftaran = ?";

try (PreparedStatement updateStatement = connection.prepareStatement(sqlUpdate))
{
    updateStatement.setString(1, idDosen);
    updateStatement.setString(2, idPendaftaran);

    int rowsAffected = updateStatement.executeUpdate();

    if (rowsAffected > 0) {
        // Notifikasi bahwa data berhasil diperbarui
        JOptionPane.showMessageDialog(null,
            "Data berhasil diperbarui.",
            "Informasi",
            JOptionPane.INFORMATION_MESSAGE);
    } else {
        // Tidak ada baris yang diperbarui
        JOptionPane.showMessageDialog(null,
            "Tidak ada data yang diperbarui.",
            "Informasi",
            JOptionPane.INFORMATION_MESSAGE);
    }
} catch (SQLException e) {
    e.printStackTrace();
    JOptionPane.showMessageDialog(null,
        "Terjadi kesalahan saat memperbarui data.",
        "Kesalahan",
        JOptionPane.ERROR_MESSAGE);
}

} else {
    // Tidak ada dosen yang dipilih
    JOptionPane.showMessageDialog(null,
        "Pilih dosen terlebih dahulu.");
}

```

```

        "Peringatan",
        JOptionPane.WARNING_MESSAGE);
    }
}

private void table2MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    int row = table2.getSelectedRow();
    String idTemp = table2.getValueAt(row, 0).toString(); // Misalkan kolom ID Tempat
    ada di kolom 0

    // Query SQL untuk mendapatkan detail dari tabel tempat_kkp
    String sql = "SELECT id_tempat, nama_perusahaan, alamat, nama_pembimbing,
    notelp, tgl_mulai, tgl_berakhir "
        + "FROM tempat_kkp "
        + "WHERE id_tempat = ?";

    try (PreparedStatement pst = connection.prepareStatement(sql)) {
        // Menetapkan parameter untuk PreparedStatement
        pst.setString(1, idTemp);

        // Menjalankan query dan memproses hasil
        try (ResultSet rs = pst.executeQuery()) {
            if (rs.next()) {
                // Menampilkan data dari tabel tempat_kkp ke dalam text fields
                tID.setText(rs.getString("id_tempat"));
                tNmPerusahaan.setText(rs.getString("nama_perusahaan"));
                tAlamat.setText(rs.getString("alamat"));
                tNmPembimbing.setText(rs.getString("nama_pembimbing"));
                tNotelp.setText(rs.getString("notelp"));
                jdcTglMulai.setDate(rs.getDate("tgl_mulai"));
                jdcTglakhir.setDate(rs.getDate("tgl_berakhir"));

                // Menonaktifkan tombol bSimpan
                bSimpan.setEnabled(false);
                bUbah.setEnabled(true);
                bHapus.setEnabled(true);
            } else {
                // Menangani kasus jika data tidak ditemukan
                JOptionPane.showMessageDialog(this,
                    "Data tidak ditemukan!",
                    "Notifikasi", JOptionPane.INFORMATION_MESSAGE);

                // Mengosongkan text fields jika data tidak ditemukan
                tID.setText("");
                tNmPerusahaan.setText("");
                tAlamat.setText("");
                tNmPembimbing.setText("");
            }
        }
    }
}

```

```

        tNotelp.setText("");
        jdcTglMulai.setDate(null);
        jdcTglakhir.setDate(null);

        // Menonaktifkan tombol bSimpan
        bSimpan.setEnabled(false);

    }

}

} catch (SQLException e) {
    // Menangani kesalahan SQL
    e.printStackTrace();
    JOptionPane.showMessageDialog(this,
        "Terjadi kesalahan saat mengambil data.",
        "Error", JOptionPane.ERROR_MESSAGE);
}

}

private void bSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    insertTempatKkp();
    refresh();
    reset();
}

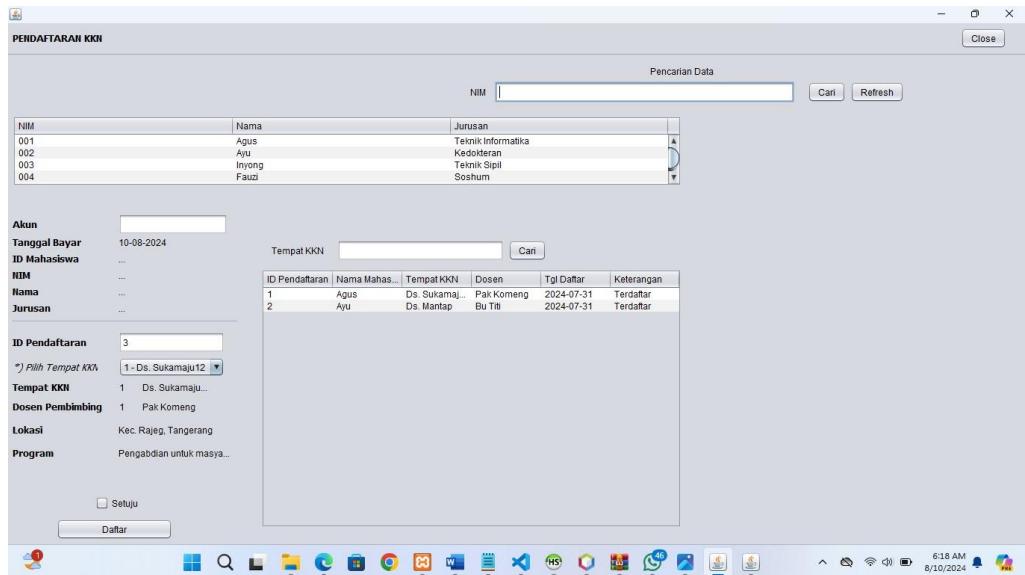
private void bBatalActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    reset();
}

private void bUbahActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    updateTempatKkp();
    refresh();
    reset();
}

private void bHapusActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    deleteTempatKkp();
    refresh();
    reset();
}

```

### 3.9 Frame daftar kkn



```

package view;
import koneksi.koneksi;
import session.session;
import model.ComboBoxItem;
import javax.swing.JComboBox;
import javax.swing.DefaultComboBoxModel;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.sql.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author fadhl
 */
public class FDaftarkkn extends javax.swing.JFrame {
    Connection connection;
    DefaultTableModel model,model2;
    // private JComboBox cbTempatKkn;

    // String nis, id_pengguna, id_spp;
    /**
     * Creates new form FBayarSPP
     */
    public FDaftarkkn() {

```

```

initComponents();
connection = koneksi.getConnection();
setTanggal();
tabelMhs();
tCari.requestFocus();
getcbTempatKkn();
generateAutoNumber();
// this.id_pengguna=id_pengguna;
// getCmbPetugas();
ambilDataTableBayar();
refresh();
akunUser();
}

public void setTanggal() {
    // Mendapatkan tanggal saat ini
    LocalDate today = LocalDate.now();

    // Menentukan format tanggal
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");

    // Mengonversi tanggal ke format yang diinginkan
    String formattedDate = today.format(formatter);

    // Menetapkan tanggal yang diformat ke komponen
    ITanggal.setText(formattedDate);
}

private void tabelMhs() {
    model = (DefaultTableModel) table.getModel();
    model.setRowCount(0); // Menghapus data lama

    String sql = "SELECT * FROM mahasiswa";

    try (Statement stat = connection.createStatement();
        ResultSet res = stat.executeQuery(sql)) {
        while (res.next()) {
            Object[] obj = new Object[3];
            obj[0] = res.getString("nim");
            obj[1] = res.getString("nama");
            obj[2] = res.getString("jurusan");
            model.addRow(obj);
        }
    } catch (SQLException err) {
        err.printStackTrace();
    }
}

private void generateAutoNumber() {
    try {
        Statement stat = connection.createStatement();

```

```

String sql = "SELECT MAX(id_pendaftaran) AS max_id FROM
pendaftaran_kkn";
ResultSet res = stat.executeQuery(sql);

if (res.next()) {
    int maxId = res.getInt("max_id");
    int newId = maxId + 1;
    tID.setText(String.valueOf(newId));
} else {
    // Handle the case where no records exist yet
    tID.setText("1");
}
} catch (SQLException e) {
    e.printStackTrace();
}
}

private void insert() {
    PreparedStatement statement = null;
    ResultSet resultSet = null;
    String sqlInsert = "INSERT INTO pendaftaran_kkn "
        + "(id_pendaftaran, id_mahasiswa, id_tempat, id_dosen, tgl_daftar,
status)"
        + " VALUES (?, ?, ?, ?, ?, ?)";
    String sqlCheck = "SELECT COUNT(*) FROM pendaftaran_kkn WHERE
id_mahasiswa = ?";

    // Ambil dan memangkas teks dari lblUsername
    String lblUsernameText = lblUsername.getText().trim();
    System.out.println("lblUsername: [" + lblUsernameText + "]");

    // Ambil 3 huruf depan dari lblUsername, dan ubah menjadi huruf kecil
    String prefix = lblUsernameText.length() >= 3 ? lblUsernameText.substring(0,
3).toLowerCase() : "";
    System.out.println("Prefix: [" + prefix + "]");

    // Ambil id_mahasiswa dari Lnim, dan ubah menjadi huruf kecil
    String idMahasiswa = Lnim.getText().trim().toLowerCase();
    System.out.println("ID Mahasiswa: [" + idMahasiswa + "]");

    // Validasi bahwa id_mahasiswa dimulai dengan prefix dari lblUsername
    if (!idMahasiswa.startsWith(prefix)) {
        JOptionPane.showMessageDialog(null,
            "ID Mahasiswa tidak sesuai.",
            "Kesalahan Validasi",
            JOptionPane.ERROR_MESSAGE);
        return;
    }

    // Cek apakah kuota sudah penuh
    String kuotaStatus = lblKuota.getText().trim();

```

```

if ("kuota penuh".equalsIgnoreCase(kuotaStatus)) {
    JOptionPane.showMessageDialog(null,
        "Kuota sudah penuh. Cari tempat KKN yang lain.",
        "Peringatan Kuota",
        JOptionPane.WARNING_MESSAGE);
    return;
}

// Cek apakah checkbox telah dipilih
if (!ckbSetuju.isSelected()) {
    JOptionPane.showMessageDialog(null,
        "Anda belum menyetujui.",
        "Kesalahan",
        JOptionPane.WARNING_MESSAGE);
    return;
}

// Periksa apakah mahasiswa sudah terdaftar
try {
    // Cek apakah mahasiswa sudah terdaftar
    statement = connection.prepareStatement(sqlCheck);
    statement.setString(1, idMahasiswa);
    resultSet = statement.executeQuery();

    if (resultSet.next() && resultSet.getInt(1) > 0) {
        // Jika mahasiswa sudah terdaftar
        JOptionPane.showMessageDialog(null,
            "Anda sudah terdaftar.",
            "Informasi",
            JOptionPane.INFORMATION_MESSAGE);
        return;
    }
}

// Jika mahasiswa belum terdaftar, lanjutkan dengan penyisipan data
statement.close(); // Tutup statement sebelumnya

statement = connection.prepareStatement(sqlInsert,
Statement.RETURN_GENERATED_KEYS);
statement.setString(1, tID.getText());
statement.setString(2, Lnim.getText()); // Pastikan penggunaan variabel yang
konsisten
statement.setString(3, Lidtempatkkn.getText());
statement.setString(4, Liddosen.getText());
statement.setDate(5, java.sql.Date.valueOf(java.time.LocalDate.now()));
statement.setString(6, ckbSetuju.isSelected() ? "Terdaftar" : "Tidak");
statement.executeUpdate();
System.out.println("Insert successful");

} catch (SQLException ex) {
    ex.printStackTrace();
} finally {

```

```

try {
    if(resultSet != null) {
        resultSet.close();
    }
    if(statement != null) {
        statement.close();
    }
} catch (SQLException ex) {
    ex.printStackTrace();
}
}

private void getcbTempatKkn() {
    cbTempatKkn.removeAllItems(); // Hapus semua item dari combo box

    // Ambil data dari tabel tempat_kkn dan dosen
    String sql = "SELECT * "
        + "FROM tempat_kkn "
        + "JOIN dosen ON tempat_kkn.id_dosen = dosen.id_dosen";

    try (Statement stat = connection.createStatement();
        ResultSet res = stat.executeQuery(sql)) {

        while (res.next()) {
            String idTempat = res.getString("id_tempat");
            String namaTempat = res.getString("nama_tempat");
            String namaDosen = res.getString("nama_dosen");
            int jumlahMahasiswa = res.getInt("jumlah_mahasiswa");

            // Format item untuk JComboBox
            String item = idTempat + " - " + namaTempat;
            cbTempatKkn.addItem(item);
        }
    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this,
            "Terjadi kesalahan saat memuat data tempat KKN.",
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}

// Method merubah tahun ke id_spp
private int getIDDosen(String nama_dosen){
    int id = 0;
    try {
        PreparedStatement st = connection.prepareStatement
            ("SELECT nama_dosen FROM dosen WHERE id_dosen",
            Statement.RETURN_GENERATED_KEYS);
        st.setString(1, nama_dosen);
        ResultSet rs = st.executeQuery();

```

```

        while (rs.next()){
            id = rs.getInt("id_dosen");
        }
        return id;
    } catch (SQLException ex) {
        Logger.getLogger(FDaftarkkn.class.getName()).log(Level.SEVERE, null, ex);
    }
    return id;
}

private void ambilDataTableBayar() {
    model2 = (DefaultTableModel) table2.getModel();
    model2.setRowCount(0);
    try {
        Statement stat = connection.createStatement();
        String sql = "SELECT * FROM pendaftaran_kkn, mahasiswa, tempat_kkn,
dosen WHERE "
                    + "pendaftaran_kkn.id_mahasiswa = mahasiswa.id_mahasiswa AND "
                    + "pendaftaran_kkn.id_tempat = tempat_kkn.id_tempat AND "
                    + "pendaftaran_kkn.id_dosen = dosen.id_dosen ";
        ResultSet res = stat.executeQuery(sql);
        while (res.next()) {
            Object[] obj = new Object[6];
            obj[0] = res.getString("id_pendaftaran");
            obj[1] = res.getString("nama");
            obj[2] = res.getString("nama_tempat");
            obj[3] = res.getString("nama_dosen");
            obj[4] = res.getString("tgl_daftar");
            obj[5] = res.getString("status");
            model2.addRow(obj);
        }
    } catch (SQLException err) {
        err.printStackTrace();
    }
}

// Method merubah nominal ke id_spp
private int getIDSPP(String nominal) {
    int id = 0;
    try {
        PreparedStatement st = connection.prepareStatement(
            "SELECT id_spp FROM spp WHERE nominal=?",
            Statement.RETURN_GENERATED_KEYS);
        st.setString(1, nominal);
        ResultSet rs = st.executeQuery();
        while (rs.next()) {
            id = rs.getInt("id_spp");
        }
        return id;
    } catch (SQLException ex) {

```

```

        Logger.getLogger(FDaftarkkn.class.getName()).log(Level.SEVERE, null,
ex);
    }
    return id;
}

private void refresh() {
model = (DefaultTableModel) table.getModel();
model.setRowCount(0);
model2 = (DefaultTableModel) table2.getModel();
model2.setRowCount(0);
tCari.setText("");
tabelMhs();
ambilDataTableBayar();
}

private void search() {
model = (DefaultTableModel) table.getModel();
model.setRowCount(0); // Menghapus data lama

String sql = "SELECT * FROM mahasiswa WHERE nim LIKE ?";

try (PreparedStatement pst = connection.prepareStatement(sql)) {
    String searchKeyword = "%" + tCari.getText().trim() + "%";
    pst.setString(1, searchKeyword);

    try (ResultSet rs = pst.executeQuery()) {
        while (rs.next()) {
            Object[] obj = new Object[3];
            obj[0] = rs.getString("nim");
            obj[1] = rs.getString("nama");
            obj[2] = rs.getString("jurusan");
            model.addRow(obj);
        }
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

private void reset() {
LidMhs.setText("...");
Lnim.setText("...");
Lnama.setText("...");
Ljurusan.setText("...");
Lidtempatkkn.setText("...");
Lnamatempat.setText("...");
Liddosen.setText("...");
Lnmdosen.setText("...");
ckbSetuju.setSelected(false);
bDaftar.setEnabled(true);
}

```

```

    }

private void akunUser() {
    // Ambil username pengguna dari SessionManager
    String username = session.getInstance().getUsername();
    // Update teks label dengan username
    lblUsername.setText(username);
}

private void search2() {
    DefaultTableModel model = (DefaultTableModel) table2.getModel();
    model2.setRowCount(0); // Menghapus data lama

    // SQL query dengan JOIN
    String sql = "SELECT pendaftaran_kkn.id_pendaftaran, mahasiswa.nama AS
    nama_mahasiswa, tempat_kkn.nama_tempat, dosen.nama_dosen AS nama_dosen,
    pendaftaran_kkn.tgl_daftar, pendaftaran_kkn.status "
        + "FROM pendaftaran_kkn "
        + "INNER JOIN mahasiswa ON pendaftaran_kkn.id_mahasiswa =
    mahasiswa.id_mahasiswa "
        + "INNER JOIN tempat_kkn ON pendaftaran_kkn.id_tempat =
    tempat_kkn.id_tempat "
        + "INNER JOIN dosen ON pendaftaran_kkn.id_dosen = dosen.id_dosen "
        + "WHERE LOWER(tempat_kkn.nama_tempat) LIKE ?";

    try (PreparedStatement pst = connection.prepareStatement(sql)) {
        // Menyiapkan parameter pencarian dengan wildcards untuk LIKE
        String searchKeyword = "%" + tCariKkn.getText().trim().toLowerCase() + "%";
        pst.setString(1, searchKeyword);

        try (ResultSet rs = pst.executeQuery()) {
            while (rs.next()) {
                Object[] obj = new Object[6];
                obj[0] = rs.getString("id_pendaftaran");
                obj[1] = rs.getString("nama_mahasiswa");
                obj[2] = rs.getString("nama_tempat");
                obj[3] = rs.getString("nama_dosen");
                obj[4] = rs.getDate("tgl_daftar");
                obj[5] = rs.getString("status");
                model.addRow(obj);
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void bCariMhsActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (!tCari.getText().trim().isEmpty()) {
        search();
    }
}

```

```

    } else {
        JOptionPane.showMessageDialog(this,
            "Masukkan NIM yang ingin dicari!",
            "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
}

private void bRefreshActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    refresh();
}

private void bDaftarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    insert();
    reset();
    refresh();
}

private void tableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    int row = table.getSelectedRow();
    String getNIM = table.getValueAt(row, 0).toString();

    // Query SQL untuk mendapatkan detail mahasiswa berdasarkan NIM
    String sql = "SELECT id_mahasiswa, nim, nama, jurusan "
        + "FROM mahasiswa "
        + "WHERE nim = ?";

    try (PreparedStatement pst = connection.prepareStatement(sql)) {
        // Menetapkan parameter untuk PreparedStatement
        pst.setString(1, getNIM);

        // Menjalankan query dan memproses hasil
        try (ResultSet rs = pst.executeQuery()) {
            if (rs.next()) {
                // Memperbarui label dengan data yang diperoleh
                LidMhs.setText(rs.getString("id_mahasiswa"));
                Lnim.setText(rs.getString("nim"));
                Lnama.setText(rs.getString("nama"));
                Ljurusan.setText(rs.getString("jurusan"));
            } else {
                // Menangani kasus jika data tidak ditemukan
                JOptionPane.showMessageDialog(this,
                    "Data mahasiswa tidak ditemukan!",
                    "Notifikasi", JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
}

// Mengosongkan label jika data tidak ditemukan
LidMhs.setText("");
Lnim.setText("");
Lnama.setText("");

```

```

        Ljurusan.setText("");
    }
}
} catch (SQLException e) {
    // Menangani kesalahan SQL
    e.printStackTrace();
    JOptionPane.showMessageDialog(this,
        "Terjadi kesalahan saat mengambil data.",
        "Error", JOptionPane.ERROR_MESSAGE);
}
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    dispose();
}

private void bCariKknActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (!tCariKkn.getText().trim().isEmpty()) {
        search2();
    } else {
        JOptionPane.showMessageDialog(this,
            "Masukkan nama tempat yang ingin dicari!",
            "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
}

private void cbTempatKknActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    // Ambil item yang dipilih
    String selectedItem = (String) cbTempatKkn.getSelectedItem();

    if (selectedItem != null) {
        // Ambil ID tempat dari item yang dipilih
        String idTempat = selectedItem.split(" - ")[0];

        // Ambil data terkait dari database
        String sql = "SELECT tempat_kkn.*, dosen.nama_dosen "
            + "FROM tempat_kkn "
            + "JOIN dosen ON tempat_kkn.id_dosen = dosen.id_dosen "
            + "WHERE tempat_kkn.id_tempat = ?";

        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, idTempat);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                String namaTempat = rs.getString("nama_tempat");
                String lokasi = rs.getString("lokasi");
                String program = rs.getString("program");
            }
        }
    }
}

```

```

String namaDosen = rs.getString("nama_dosen");
int jumlahMahasiswa = rs.getInt("jumlah_mahasiswa");

// Update label dengan data dari database
Lidtempatkkn.setText(idTempat);
Lnamatempat.setText(namaTempat);
Liddosen.setText(rs.getString("id_dosen"));
Lnmdosen.setText(namaDosen);
lblLokasi.setText(lokasi);
lblProgram.setText(program);

// Tambahkan tanda kuota penuh jika diperlukan
if (jumlahMahasiswa >= 4) {
    lblKuota.setText("Kuota Penuh");
} else {
    lblKuota.setText("");
}
}

} catch (SQLException e) {
e.printStackTrace();
JOptionPane.showMessageDialog(this,
    "Terjadi kesalahan saat memuat detail tempat KKN.",
    "Error", JOptionPane.ERROR_MESSAGE);
}
}
}

```

### 3.11 Frame daftar kkp

NIM	Nama	Jurusan
001	Agus	Teknik Informatika
002	Ayu	Kedokteran
003	Inyong	Teknik Sipil
004	Fauzi	Soshum

ID Pendaftaran	Nama Mahasiswa	Tempat KKP	Tgl Daftar
2	Ujang	PT. Ching Luh	2024-08-04
1	Uta	PT. Stanley	2024-08-04

ID Pendaftaran	Nama Dosen	Status
2	Bu Jihyo	approve
1	Bu Jihyo	approve

\*) Isi Form KKP

ID Pendaftaran	<input type="text" value="3"/>	Nama Pembimbing Perusahaan	<input type="text"/>
ID Tempat	<input type="text" value="3"/>	No Telepon / Handphone PP	<input type="text"/>
Nama Perusahaan	<input type="text"/>	Tanggal Mulai	<input type="text"/> <input type="button" value="..."/>
Alamat	<input type="text"/>	Tanggal Berakhir	<input type="text"/> <input type="button" value="..."/>

\*) Jika KKP Disetujui maka akan muncul 'Approve' beserta nama Dosen Pembimbing, Jika tidak 'Not Approve'

....

OK Cancel Daftar

```

package view;
import koneksi.koneksi;
import session.session;
import java.sql.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

```

/\*\*

```

/*
 * @author fadhl
 */
public class FDaftarkkp extends javax.swing.JFrame {
    Connection connection;
    DefaultTableModel model,model2,model3;
    /**
     * Creates new form FDaftarkkp
     */
    public FDaftarkkp() {
        initComponents();
        connection = koneksi.getConnection();
        setTanggal();
        tabelMhs();
        tabelDaftarkkp();
        tabelkkpdetail();
        tCari.requestFocus();
        generateAutoNumber();
        autoNumberTempatkkp();
        // refresh();
        akunUser();
        bDaftar.setEnabled(false);
    }

    public void setTanggal() {
        // Mendapatkan tanggal saat ini
        LocalDate today = LocalDate.now();

        // Menentukan format tanggal
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");

        // Mengonversi tanggal ke format yang diinginkan
        String formattedDate = today.format(formatter);

        // Menetapkan tanggal yang diformat ke komponen
        ITanggal.setText(formattedDate);
    }

    private void tabelMhs() {
        model = (DefaultTableModel) table.getModel();
        model.setRowCount(0); // Menghapus data lama

        String sql = "SELECT * FROM mahasiswa";

        try (Statement stat = connection.createStatement();
             ResultSet res = stat.executeQuery(sql)) {
            while (res.next()) {
                Object[] obj = new Object[3];
                obj[0] = res.getString("nim");

```

```

        obj[1] = res.getString("nama");
        obj[2] = res.getString("jurusan");
        model.addRow(obj);
    }
} catch (SQLException err) {
    err.printStackTrace();
}
}

private void akunUser() {
    // Ambil username pengguna dari SessionManager
    String username = session.getInstance().getUsername();
    // Update teks label dengan username
    lblUsername.setText(username);
}

private void search() {
    model = (DefaultTableModel) table.getModel();
    model.setRowCount(0); // Menghapus data lama

    String sql = "SELECT * FROM mahasiswa WHERE nim LIKE ?";

    try (PreparedStatement pst = connection.prepareStatement(sql)) {
        String searchKeyword = "%" + tCari.getText().trim() + "%";
        pst.setString(1, searchKeyword);

        try (ResultSet rs = pst.executeQuery()) {
            while (rs.next()) {
                Object[] obj = new Object[3];
                obj[0] = rs.getString("nim");
                obj[1] = rs.getString("nama");
                obj[2] = rs.getString("jurusan");
                model.addRow(obj);
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void refresh() {
    model = (DefaultTableModel) table.getModel();
    model.setRowCount(0);
    model2 = (DefaultTableModel) table2.getModel();
    model2.setRowCount(0);
    tCari.setText("");
    tabelMhs();
    tabelDaftarkkp();
    tabelkkpdetail();
}

```

```

}

private void tabelDaftarkkp() {
    model2 = (DefaultTableModel) table2.getModel();
    model2.setRowCount(0);
    try {
        Statement stat = connection.createStatement();
        String sql = "SELECT * FROM pendaftaran_kkp, mahasiswa, tempat_kkp
WHERE "
        + "pendaftaran_kkp.id_mahasiswa = mahasiswa.id_mahasiswa AND "
        + "pendaftaran_kkp.id_tempat = tempat_kkp.id_tempat "
        + "ORDER BY pendaftaran_kkp.id_mahasiswa DESC";

        ResultSet res = stat.executeQuery(sql);
        while (res.next()) {
            Object[] obj = new Object[4];
            obj[0] = res.getString("id_pendaftaran");
            obj[1] = res.getString("nama");
            obj[2] = res.getString("nama_perusahaan");
            obj[3] = res.getString("tgl_daftar");
            model2.addRow(obj);
        }
        checkMahasiswaStatus();
    } catch (SQLException err) {
        err.printStackTrace();
    }
}

private void tabelkkpdetail() {
    model3 = (DefaultTableModel) tabel3.getModel();
    model3.setRowCount(0);
    try {
        Statement stat = connection.createStatement();
        String sql = "SELECT pendaftaran_kkpdetail.id_pendaftaran, dosen.nama_dosen,
pendaftaran_kkpdetail.status "
        + "FROM pendaftaran_kkpdetail "
        + "INNER JOIN dosen ON pendaftaran_kkpdetail.id_dosen =
dosen.id_dosen "
        + "ORDER BY pendaftaran_kkpdetail.id_pendaftaran DESC";

        ResultSet res = stat.executeQuery(sql);
        while (res.next()) {
            Object[] obj = new Object[3];
            obj[0] = res.getString("id_pendaftaran");
            obj[1] = res.getString("nama_dosen");
            obj[2] = res.getString("status");
            model3.addRow(obj);
        }
        updateLstatusIfTableEmpty();
    }
}

```

```

        } catch (SQLException err) {
            err.printStackTrace();
        }
    }

private void insertPendaftaranKkp() {
    PreparedStatement statement = null;
    ResultSet resultSet = null;

    // SQL untuk memasukkan data ke tabel pendaftaran_kkp
    String sqlInsert = "INSERT INTO pendaftaran_kkp "
        + "(id_pendaftaran, id_mahasiswa, id_tempat, tgl_daftar) "
        + "VALUES (?, ?, ?, ?)";

    // SQL untuk memeriksa apakah mahasiswa sudah terdaftar di tempat tersebut
    String sqlCheck = "SELECT COUNT(*) FROM pendaftaran_kkp "
        + "WHERE id_mahasiswa = ? AND id_tempat = ?";

    // Ambil data dari GUI
    String idPendaftaran = tID.getText().trim();
    String idMahasiswa = LidMhs.getText().trim();
    String idTempat = tIDtempat.getText().trim();
    java.sql.Date tglDaftar = java.sql.Date.valueOf(java.time.LocalDate.now());

    // Ambil dan memangkas teks dari lblUsername
    String lblUsernameText = lblUsername.getText().trim();
    // Ambil 3 huruf depan dari lblUsername, dan ubah menjadi huruf kecil
    String prefix = lblUsernameText.length() >= 3 ? lblUsernameText.substring(0,
3).toLowerCase() : "";

    // Ambil id_mahasiswa dari Lnim
    String nim = Lnim.getText().trim();
    String nimPrefix = nim.length() >= 3 ? nim.substring(0, 3).toLowerCase() : "";

    // Validasi ID Pendaftaran, ID Mahasiswa, dan ID Tempat tidak kosong
    if (idPendaftaran.isEmpty() || idMahasiswa.isEmpty() || idTempat.isEmpty()) {
        JOptionPane.showMessageDialog(null,
            "ID Pendaftaran, ID Mahasiswa, dan ID Tempat tidak boleh kosong.",
            "Kesalahan Validasi",
            JOptionPane.ERROR_MESSAGE);
        return;
    }

    // Validasi bahwa NIM dimulai dengan prefix dari lblUsername
    if (!nimPrefix.equals(prefix)) {
        JOptionPane.showMessageDialog(null,
            "NIM tidak sesuai dengan tiga huruf pertama dari username.",
            "Kesalahan Validasi",
            JOptionPane.ERROR_MESSAGE);
    }
}

```

```

        return;
    }

try (PreparedStatement checkStatement = connection.prepareStatement(sqlCheck)) {
    // Cek apakah mahasiswa sudah terdaftar di tempat tersebut
    checkStatement.setString(1, idMahasiswa);
    checkStatement.setString(2, idTempat);
    try (ResultSet checkResultSet = checkStatement.executeQuery()) {
        if (checkResultSet.next() && checkResultSet.getInt(1) > 0) {
            // Jika mahasiswa sudah terdaftar di tempat tersebut
            JOptionPane.showMessageDialog(null,
                "Mahasiswa sudah terdaftar di tempat ini.",
                "Informasi",
                JOptionPane.INFORMATION_MESSAGE);
            return;
        }
    }
}

// Jika mahasiswa belum terdaftar, lanjutkan dengan penyisipan data
try (PreparedStatement insertStatement = connection.prepareStatement(sqlInsert)) {
    insertStatement.setString(1, idPendaftaran); // ID Pendaftaran
    insertStatement.setString(2, idMahasiswa); // ID Mahasiswa
    insertStatement.setString(3, idTempat); // ID Tempat
    insertStatement.setDate(4, tglDaftar); // Tanggal Daftar
    insertStatement.executeUpdate();

    // Notifikasi bahwa data berhasil disimpan
    JOptionPane.showMessageDialog(null,
        "Data berhasil disimpan.",
        "Informasi",
        JOptionPane.INFORMATION_MESSAGE);
    System.out.println("Insert successful");
}

} catch (SQLException ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(null,
        "Terjadi kesalahan saat menyimpan data.",
        "Kesalahan",
        JOptionPane.ERROR_MESSAGE);
}
}

private void insertTempatKkp() {
    PreparedStatement statement = null;
    ResultSet resultSet = null;

    // SQL untuk memasukkan data ke tabel tempat_kkp
    String sqlInsert = "INSERT INTO tempat_kkp "

```

```

        + "(id_tempat, nama_perusahaan, alamat, nama_pembimbing, notelp,
tgl_mulai, tgl_berakhir) "
        + "VALUES (?, ?, ?, ?, ?, ?, ?, ?);"

// SQL untuk memeriksa apakah ID Tempat sudah ada
String sqlCheck = "SELECT COUNT(*) FROM tempat_kkp WHERE id_tempat =
?";

// Ambil data dari GUI
String idTempat = tIDtempat.getText().trim();
String namaPerusahaan = tNamaPerusahaan.getText().trim();
String alamat = tAlamat.getText().trim();
String namaPembimbing = tNmPembimbing.getText().trim();
String notelp = tNotelp.getText().trim();
java.sql.Date tglMulai = new java.sql.Date(jdcMulai.getDate().getTime());
java.sql.Date tglBerakhir = new java.sql.Date(jdcAkhir.getDate().getTime());

// Validasi ID Tempat tidak kosong
if (idTempat.isEmpty()) {
    JOptionPane.showMessageDialog(null,
        "ID Tempat tidak boleh kosong.",
        "Kesalahan Validasi",
        JOptionPane.ERROR_MESSAGE);
    return;
}

try {
    // Cek apakah ID Tempat sudah ada
    try (PreparedStatement checkStatement =
connection.prepareStatement(sqlCheck)) {
        checkStatement.setString(1, idTempat);
        try (ResultSet checkResultSet = checkStatement.executeQuery()) {
            if (checkResultSet.next() && checkResultSet.getInt(1) > 0) {
                // Jika ID Tempat sudah ada
                JOptionPane.showMessageDialog(null,
                    "ID Tempat sudah ada.",
                    "Informasi",
                    JOptionPane.INFORMATION_MESSAGE);
                return;
            }
        }
    }
}

// Jika ID Tempat belum ada, lanjutkan dengan penyisipan data
try (PreparedStatement insertStatement = connection.prepareStatement(sqlInsert))
{
    insertStatement.setString(1, idTempat); // ID Tempat
    insertStatement.setString(2, namaPerusahaan); // Nama Perusahaan
    insertStatement.setString(3, alamat); // Alamat
    insertStatement.setString(4, namaPembimbing); // Nama Pembimbing
}

```

```

        insertStatement.setString(5, notelp); // No Telp
        insertStatement.setDate(6, tglMulai); // Tanggal Mulai
        insertStatement.setDate(7, tglBerakhir); // Tanggal Berakhir
        insertStatement.executeUpdate();

        // Notifikasi bahwa data berhasil disimpan
        JOptionPane.showMessageDialog(null,
            "Data berhasil disimpan.",
            "Informasi",
            JOptionPane.INFORMATION_MESSAGE);
        System.out.println("Insert successful");

        // Aktifkan tombol bDaftar jika data berhasil disimpan
        bDaftar.setEnabled(true);
    }

} catch (SQLException ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(null,
        "Terjadi kesalahan saat menyimpan data.",
        "Kesalahan",
        JOptionPane.ERROR_MESSAGE);

    // Nonaktifkan tombol bDaftar jika terjadi kesalahan
    bDaftar.setEnabled(false);
} finally {
    try {
        if (resultSet != null) {
            resultSet.close();
        }
        if (statement != null) {
            statement.close();
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}
}

private void generateAutoNumber() {
    try {
        Statement stat = connection.createStatement();
        String sql = "SELECT MAX(id_pendaftaran) AS max_id FROM
pendaftaran_kkp";
        ResultSet res = stat.executeQuery(sql);

        if (res.next()) {
            int maxId = res.getInt("max_id");
            int newId = maxId + 1;

```

```

        tID.setText(String.valueOf(newId));
    } else {
        // Handle the case where no records exist yet
        tID.setText("1");
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

private void autoNumberTempatKkp() {
try {
    Statement stat = connection.createStatement();
    String sql = "SELECT MAX(id_tempat) AS max_id FROM tempat_kkp";
    ResultSet res = stat.executeQuery(sql);

    if (res.next()) {
        int maxId = res.getInt("max_id");
        int newId = maxId + 1;
        tIDtempat.setText(String.valueOf(newId));
    } else {
        // Handle the case where no records exist yet
        tIDtempat.setText("1");
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

private void reset() {
    LidMhs.setText("...");
    Lnim.setText("...");
    Lnama.setText("...");
    Ljurusan.setText("..");
    bDaftar.setEnabled(false);
    generateAutoNumber();
    autoNumberTempatKkp();
    tNamaPerusahaan.setText("");
    tAlamat.setText("");
    tNmPembimbing.setText("");
    tNotelp.setText("");
    tNotelp.setText("");
}

private void updateLstatusIfTableEmpty() {
// Pastikan model tabel yang digunakan adalah DefaultTableModel
DefaultTableModel model3 = (DefaultTableModel) tabel3.getModel();

// Cek apakah tabel kosong

```

```

        if (model3.getRowCount() == 0) {
            // Jika tabel kosong, set Lstatus menjadi 'Not Approved'
            Lstatus.setText("Not Approved");
        }
    }

private void checkMahasiswaStatus() {
    String idMahasiswa = LidMhs.getText().trim();
    boolean isRegistered = false;

    // Dapatkan model tabel dari tabel2
    DefaultTableModel model = (DefaultTableModel) table2.getModel();

    // Iterasi melalui baris tabel untuk mencari ID Mahasiswa
    for (int i = 0; i < model.getRowCount(); i++) {
        String mahasiswaInTable = model.getValueAt(i, 0).toString(); // Sesuaikan
        dengan kolom ID Mahasiswa
        if (mahasiswaInTable.equals(idMahasiswa)) {
            isRegistered = true;
            break;
        }
    }

    // Aktifkan atau nonaktifkan tombol berdasarkan status
    bOk.setEnabled(!isRegistered);
    bDaftar.setEnabled(!isRegistered);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    dispose();
}

private void bCariMhsActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (!tCari.getText().trim().isEmpty()) {
        search();
    } else {
        JOptionPane.showMessageDialog(this,
            "Masukkan NIM yang ingin dicari!",
            "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
}

private void bRefreshActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    refresh();
}

```

```

private void tableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    int row = table.getSelectedRow();
    String getNIM = table.getValueAt(row, 0).toString();

    // Query SQL untuk mendapatkan detail mahasiswa berdasarkan NIM
    String sql = "SELECT id_mahasiswa, nim, nama, jurusan "
    + "FROM mahasiswa "
    + "WHERE nim = ?";

    try (PreparedStatement pst = connection.prepareStatement(sql)) {
        // Menetapkan parameter untuk PreparedStatement
        pst.setString(1, getNIM);

        // Menjalankan query dan memproses hasil
        try (ResultSet rs = pst.executeQuery()) {
            if (rs.next()) {
                // Memperbarui label dengan data yang diperoleh
                LidMhs.setText(rs.getString("id_mahasiswa"));
                Lnim.setText(rs.getString("nim"));
                Lnama.setText(rs.getString("nama"));
                Ljurusan.setText(rs.getString("jurusan"));
            } else {
                // Menangani kasus jika data tidak ditemukan
                JOptionPane.showMessageDialog(this,
                    "Data mahasiswa tidak ditemukan!",
                    "Notifikasi", JOptionPane.INFORMATION_MESSAGE);

                // Mengosongkan label jika data tidak ditemukan
                LidMhs.setText("");
                Lnim.setText("");
                Lnama.setText("");
                Ljurusan.setText("");
            }
        }
    } catch (SQLException e) {
        // Menangani kesalahan SQL
        e.printStackTrace();
        JOptionPane.showMessageDialog(this,
            "Terjadi kesalahan saat mengambil data.",
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}

private void bDaftarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    insertPendaftaranKkp();
    refresh();
}

```

```

private void bOkActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    insertTempatKkp();
    refresh();
}

private void bCancelActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    reset();
}

```

### 3.12 Evaluasi KKN

The screenshot shows a Java Swing application window titled "Evaluasi / Penilaian KKN". The window has a title bar with a close button. Below the title bar is a search bar labeled "Pencarian Data" with a text input field and three buttons: "Cari" (Search), "Refresh", and "Cetak Mahasiswa Terdaftar" (Print Registered Students). A table below the search bar displays student information:

ID Pendaftaran	NIM	Nama	Jurusan	Tempat KKN
1	001	Agus	Teknik Informatika	Ds. Sukamaju12
2	002	Ayu	Kedokteran	Ds. Mantap
3	003	Inyong	Teknik Sipil	Ds. Sukamaju12

On the left side of the window, there is a vertical stack of input fields and labels:

- ID Evaluasi: 3
- Tanggal: 29-08-2024
- ID Pendaftaran: ...
- NIM: ...
- Nama: ...
- Jurusan: ...
- Tempat KKN: ...
- Nilai:

On the right side, there is a table showing evaluation details:

Tanggal	ID Pendaft...	NIM	Nama Mah...	Tempat K...	Nilai	Keterangan
2024-07-31	1	001	Agus	Ds. Suka...	90	A (Sangat ...)
2024-07-31	2	002	Ayu	Ds. Mantap	80	A (Sangat ...)

At the bottom of the window are two buttons: "Submit" and "Cetak".

```

package view;
import koneksi.koneksi;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

```

```

/**
 *
 * @author fadhl
 */
public class FEvaluasikkn extends javax.swing.JFrame {
    Connection connection;
    DefaultTableModel model,model2;
    /**
     * Creates new form FBayarSPP
     */
    public FEvaluasikkn() {
        initComponents();
        connection = koneksi.getConnection();
        setTanggal();
        getDataTable();
        tCari.requestFocus();
        generateAutoNumber();
        ambilDataTableBayar();
        refresh();
    }

    public void setTanggal() {
        // Mendapatkan tanggal saat ini
        LocalDate today = LocalDate.now();

        // Menentukan format tanggal
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");

        // Mengonversi tanggal ke format yang diinginkan
        String formattedDate = today.format(formatter);

        // Menetapkan tanggal yang diformat ke komponen
        iTanggal.setText(formattedDate);
    }

    private void getDataTable() {
        model = (DefaultTableModel) table.getModel();
        model.setRowCount(0);
        try {
            Statement stat = connection.createStatement();
            String sql = "SELECT * FROM pendaftaran_kkn,mahasiswa,tempat_kkn "
                    + "WHERE pendaftaran_kkn.id_mahasiswa=mahasiswa.id_mahasiswa AND "
                    + "pendaftaran_kkn.id_tempat=tempat_kkn.id_tempat ";
            ResultSet res = stat.executeQuery(sql);
            while (res.next()) {
                Object[] obj = new Object[5];
                obj[0] = res.getString("id_pendaftaran");
                obj[1] = res.getString("nim");
                obj[2] = res.getString("nama");
                obj[3] = res.getString("jurusan");
            }
        }
    }
}

```

```

        obj[4] = res.getString("nama_tempat");
        model.addRow(obj);
    }
} catch (SQLException err) {
    err.printStackTrace();
}
}

private void insert() {
    PreparedStatement statement = null;
    String sql = "INSERT INTO evaluasi "
        + "(id_evaluasi, tgl, id_pendaftaran,id_mahasiswa, nilai, keterangan)"
        + " VALUES (?, ?, ?, ?, ?, ?)";
    try {
        statement = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, LidEval.getText());
        statement.setString(2, java.sql.Date.valueOf(java.time.LocalDate.now()).toString());
        statement.setString(3, LIDPendaftaran.getText());
        statement.setString(4, LidMhs.getText());
        statement.setString(5, tNilai.getText());
        statement.setString(6, LKet.getText());
        statement.executeUpdate();
    } catch (SQLException ex) {
        ex.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

private void ambilDataTableBayar() {
    model2 = (DefaultTableModel) table2.getModel();
    model2.setRowCount(0);
    try {
        Statement stat = connection.createStatement();
        String sql = "SELECT * FROM pendaftaran_kkn, mahasiswa, tempat_kkn, evaluasi
WHERE "
            + "evaluasi.id_pendaftaran = pendaftaran_kkn.id_pendaftaran AND "
            + "pendaftaran_kkn.id_tempat = tempat_kkn.id_tempat AND "
            + "pendaftaran_kkn.id_mahasiswa = mahasiswa.id_mahasiswa ";
        ResultSet res = stat.executeQuery(sql);
        while (res.next()) {
            Object[] obj = new Object[7];
            obj[0] = res.getString("tgl_daftar");
            obj[1] = res.getString("id_pendaftaran");
            obj[2] = res.getString("nim");

```

```

        obj[3] = res.getString("nama");
        obj[4] = res.getString("nama_tempat");
        obj[5] = res.getString("nilai");
        obj[6] = res.getString("keterangan");
        model2.addRow(obj);
    }
} catch (SQLException err) {
    err.printStackTrace();
}
}

private void generateAutoNumber() {
try {
    Statement stat = connection.createStatement();
    String sql = "SELECT MAX(id_evaluasi) AS max_id FROM evaluasi";
    ResultSet res = stat.executeQuery(sql);

    if (res.next()) {
        int maxId = res.getInt("max_id");
        int newId = maxId + 1;
        LidEval.setText(String.valueOf(newId));
    } else {
        // Handle the case where no records exist yet
        LidEval.setText("1");
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

private void refresh() {
model = (DefaultTableModel) table.getModel();
model.setRowCount(0);
model2 = (DefaultTableModel) table2.getModel();
model2.setRowCount(0);
tCari.setText("");
getDataTable();
ambilDataTableBayar();
LKet.setText(..");
generateAutoNumber();
}

private void search() {
DefaultTableModel model = (DefaultTableModel) table.getModel();
model.setRowCount(0); // Menghapus data lama

// SQL query dengan JOIN
String sql = "SELECT pendaftaran_kkn.id_pendaftaran, mahasiswa.nim,
mahasiswa.nama, mahasiswa.jurusan, tempat_kkn.nama_tempat "
+ "FROM pendaftaran_kkn "

```

```

+ "INNER JOIN mahasiswa ON pendaftaran_kkn.id_mahasiswa =
mahasiswa.id_mahasiswa "
+ "INNER JOIN tempat_kkn ON pendaftaran_kkn.id_tempat =
tempat_kkn.id_tempat "
+ "WHERE LOWER(tempat_kkn.nama_tempat) LIKE ?";

try (PreparedStatement pst = connection.prepareStatement(sql)) {
    // Menyiapkan parameter pencarian dengan wildcards untuk LIKE
    String searchKeyword = "%" + tCari.getText().trim().toLowerCase() + "%";
    pst.setString(1, searchKeyword);

    try (ResultSet rs = pst.executeQuery()) {
        while (rs.next()) {
            Object[] obj = new Object[5];
            obj[0] = rs.getString("id_pendaftaran");
            obj[1] = rs.getString("nim");
            obj[2] = rs.getString("nama");
            obj[3] = rs.getString("jurusan");
            obj[4] = rs.getString("nama_tempat");
            model.addRow(obj);
        }
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

private void reset() {
    LIDPendaftaran.setText("...");
    Lnim.setText("...");
    Lnama.setText("...");
    Ljurusan.setText("...");
    Lidtempatkkn.setText("...");
    Lnamatempat.setText("...");
    tNilai.setText(" ");
    LKet.setText("..");
    bSubmit.setEnabled(true);
}

private void bCariActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (!tCari.getText().trim().isEmpty()) {
        search(); // Pastikan search() adalah metode yang benar
    } else {
        JOptionPane.showMessageDialog(this,
            "Masukkan nama tempat yang ingin dicari!",
            "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
}

```

```

private void bRefreshActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    refresh();
}

private void bSubmitActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    insert();
    reset();
    refresh();
}

private void tableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    String getNIM = table.getValueAt(table.getSelectedRow(), 0).toString();
    String getNama = table.getValueAt(table.getSelectedRow(), 1).toString();
    // get data dari Table berdasarkan NIS
    try{
        String sql = "SELECT * FROM pendaftaran_kkn,mahasiswa,tempat_kkn "
            + "WHERE pendaftaran_kkn.id_mahasiswa=mahasiswa.id_mahasiswa AND "
            + "pendaftaran_kkn.id_tempat=tempat_kkn.id_tempat AND id_pendaftaran =
        "+getNIM+"";
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(sql);
        while(rs.next()) {
            LIDPendaftaran.setText(rs.getString("id_pendaftaran"));
            LidMhs.setText(rs.getString("id_mahasiswa"));
            Lnim.setText(rs.getString("nim"));
            Lnama.setText(rs.getString("nama"));
            Ljurusan.setText(rs.getString("jurusan"));
            Lidtempatkkn.setText(rs.getString("id_tempat"));
            Lnamatempat.setText(rs.getString("nama_tempat"));
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

private void tNilaiActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void tNilaiKeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    String nilaiStr = tNilai.getText();
    int nilai = 0;

    try {
        nilai = Integer.parseInt(nilaiStr);
    } catch (NumberFormatException e) {

```

```

LKet.setText("Nilai harus berupa angka.");
return;
}

String keterangan;
if (nilai >= 80 && nilai <= 100) {
    keterangan = "A (Sangat Baik)";
} else if (nilai >= 70 && nilai < 80) {
    keterangan = "B (Baik)";
} else if (nilai >= 60 && nilai < 70) {
    keterangan = "C (Sangat Cukup)";
} else if (nilai >= 50 && nilai < 60) {
    keterangan = "D (Cukup)";
} else if (nilai >= 0 && nilai < 50) {
    keterangan = "E (Ulang Tahun Depan)";
} else {
    keterangan = "Nilai di luar jangkauan.";
}

LKet.setText(keterangan);
}

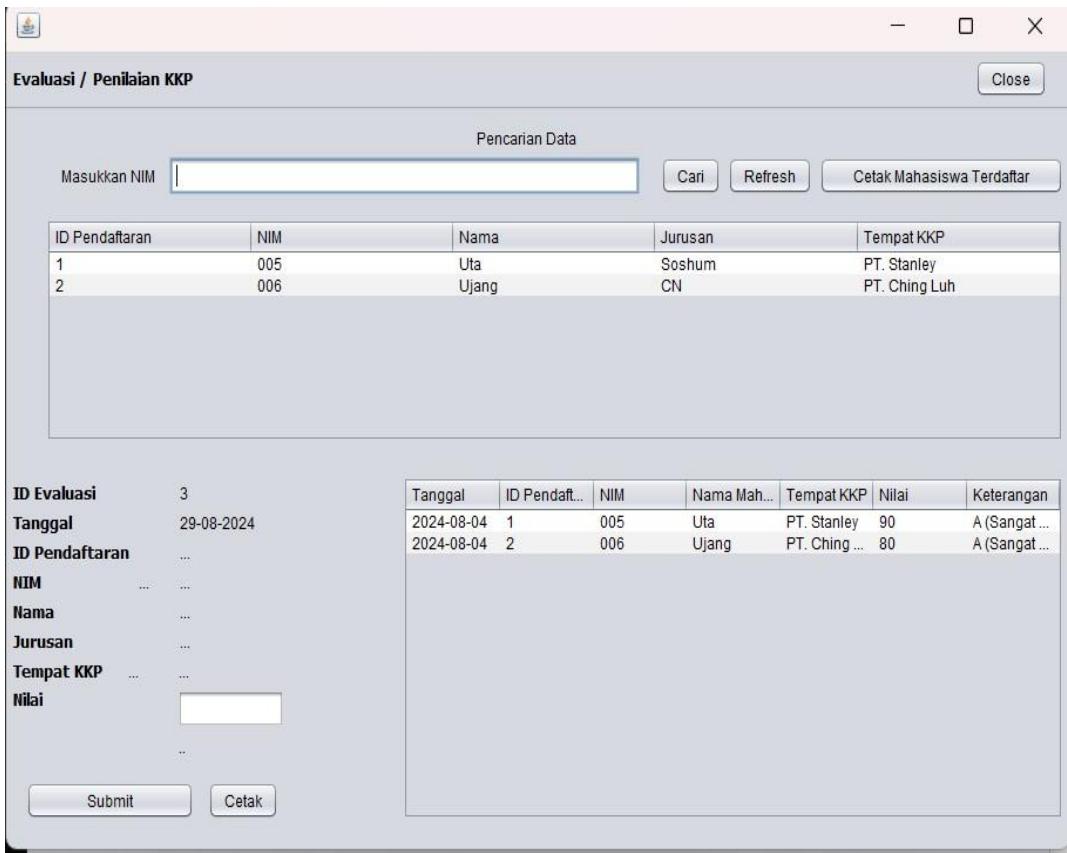
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    dispose();
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JPrmkkn F = new JPrmkkn();
    F.setVisible(true);
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JPrmNilaikkn rkkn = new JPrmNilaikkn();
    rkkn.setVisible(true);
}

```

### 3.13 Evaluasi KKP



```

package view;
import koneksi.koneksi;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author fadhl
 */
public class FEvaluasikkp extends javax.swing.JFrame {
    Connection connection;
    DefaultTableModel model,model2;
//  String nis, id_pengguna, id_spp;
    /**
     * Creates new form FBayarSPP
     */
    
```

```

public FEvaluasikkp() {
    initComponents();
    connection = koneksi.getConnection();
    setTanggal();
    getDataTable();
    tCari.requestFocus();
    generateAutoNumber();
    ambilDataTableBayar();
    refresh();
}

public void setTanggal() {
    // Mendapatkan tanggal saat ini
    LocalDate today = LocalDate.now();

    // Menentukan format tanggal
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");

    // Mengonversi tanggal ke format yang diinginkan
    String formattedDate = today.format(formatter);

    // Menetapkan tanggal yang diformat ke komponen
    iTanggal.setText(formattedDate);
}

private void getDataTable() {
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    model.setRowCount(0);
    try {
        Statement stat = connection.createStatement();
        String sql = "SELECT * FROM pendaftaran_kkp "
            + "INNER JOIN mahasiswa ON pendaftaran_kkp.id_mahasiswa = "
            + "mahasiswa.id_mahasiswa "
            + "INNER JOIN tempat_kkp ON pendaftaran_kkp.id_tempat = "
            + "tempat_kkp.id_tempat";
        ResultSet res = stat.executeQuery(sql);
        while (res.next()) {
            Object[] obj = new Object[5];
            obj[0] = res.getString("id_pendaftaran");
            obj[1] = res.getString("nim");
            obj[2] = res.getString("nama");
            obj[3] = res.getString("jurusan");
            obj[4] = res.getString("nama_perusahaan");
            model.addRow(obj);
        }
    } catch (SQLException err) {
        err.printStackTrace();
    }
}

```

```

private void insert() {
    PreparedStatement statement = null;
    String sql = "INSERT INTO evaluasi_kkp "
        + "(id_evaluasi, tgl, id_pendaftaran,id_mahasiswa, nilai, keterangan)"
        + " VALUES (?, ?, ?, ?, ?, ?)";
    try {
        statement = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
        // statement.setInt(1, getIDPengguna(cmbPengguna.getSelectedItem().toString()));
        // statement.setString(1, id_pengguna)
        statement.setString(1, LidEval.getText());
        statement.setString(2, java.sql.Date.valueOf(java.time.LocalDate.now()).toString());
        statement.setString(3, LIDPendaftaran.getText());
        statement.setString(4, LidMhs.getText());
        statement.setString(5, tNilai.getText());
        statement.setString(6, LKet.getText());
        statement.executeUpdate();
    } catch (SQLException ex) {
        ex.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

private void ambilDataTableBayar() {
    model2 = (DefaultTableModel) table2.getModel();
    model2.setRowCount(0);
    try {
        Statement stat = connection.createStatement();
        String sql = "SELECT * FROM pendaftaran_kkp, mahasiswa, tempat_kkp,
evaluasi_kkp WHERE "
            + "evaluasi_kkp.id_pendaftaran = pendaftaran_kkp.id_pendaftaran AND "
            + "pendaftaran_kkp.id_tempat = tempat_kkp.id_tempat AND "
            + "pendaftaran_kkp.id_mahasiswa = mahasiswa.id_mahasiswa ";
        ResultSet res = stat.executeQuery(sql);
        while (res.next()) {
            Object[] obj = new Object[7];
            obj[0] = res.getString("tgl_daftar");
            obj[1] = res.getString("id_pendaftaran");
            obj[2] = res.getString("nim");
            obj[3] = res.getString("nama");
            obj[4] = res.getString("nama_perusahaan");
            obj[5] = res.getString("nilai");
            obj[6] = res.getString("keterangan");
            model2.addRow(obj);
        }
    }
}

```

```

        }
    } catch (SQLException err) {
        err.printStackTrace();
    }
}

private void generateAutoNumber() {
    try {
        Statement stat = connection.createStatement();
        String sql = "SELECT MAX(id_evaluasi) AS max_id FROM evaluasi";
        ResultSet res = stat.executeQuery(sql);

        if (res.next()) {
            int maxId = res.getInt("max_id");
            int newId = maxId + 1;
            LidEval.setText(String.valueOf(newId));
        } else {
            // Handle the case where no records exist yet
            LidEval.setText("1");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void refresh() {
    model = (DefaultTableModel) table.getModel();
    model.setRowCount(0);
    model2 = (DefaultTableModel) table2.getModel();
    model2.setRowCount(0);
    tCari.setText("");
    getDataTable();
    ambilDataTableBayar();
    LKet.setText(..");
    generateAutoNumber();
}

private void search() {
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    model.setRowCount(0); // Menghapus data lama

    // SQL query dengan JOIN untuk pencarian berdasarkan nim dan nama mahasiswa
    String sql = "SELECT pendaftaran_kkp.id_pendaftaran, mahasiswa.nim,
    mahasiswa.nama, mahasiswa.jurusan, tempat_kkp.nama_perusahaan "
    + "FROM pendaftaran_kkp "
    + "INNER JOIN mahasiswa ON pendaftaran_kkp.id_mahasiswa =
    mahasiswa.id_mahasiswa "
    + "INNER JOIN tempat_kkp ON pendaftaran_kkp.id_tempat =
    tempat_kkp.id_tempat "
}

```

```

+ "WHERE LOWER(mahasiswa.nim) LIKE ? OR LOWER(mahasiswa.nama)
LIKE ?";

try (PreparedStatement pst = connection.prepareStatement(sql)) {
    // Menyiapkan parameter pencarian dengan wildcards untuk LIKE
    String searchKeyword = "%" + tCari.getText().trim().toLowerCase() + "%";
    pst.setString(1, searchKeyword); // Pencarian berdasarkan nim
    pst.setString(2, searchKeyword); // Pencarian berdasarkan nama

    try (ResultSet rs = pst.executeQuery()) {
        while (rs.next()) {
            Object[] obj = new Object[5];
            obj[0] = rs.getString("id_pendaftaran");
            obj[1] = rs.getString("nim");
            obj[2] = rs.getString("nama");
            obj[3] = rs.getString("jurusan");
            obj[4] = rs.getString("nama_perusahaan");
            model.addRow(obj);
        }
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

private void reset() {
    LIDPendaftaran.setText("...");
    Lnim.setText("...");
    Lnama.setText("...");
    Ljurusan.setText("..");
    Lidtempatkp.setText(..);
    Lnamatempat.setText("..");
    tNilai.setText(" ");
    LKet.setText("..");
    bSubmit.setEnabled(true);
}

private void bCariActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (!tCari.getText().trim().isEmpty()) {
        search(); // Pastikan search() adalah metode yang benar
    } else {
        JOptionPane.showMessageDialog(this,
            "Masukkan nama tempat yang ingin dicari!",
            "Notifikasi", JOptionPane.WARNING_MESSAGE);
    }
}

private void bRefreshActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

```

```

        refresh();
    }

private void bSubmitActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    insert();
    reset();
    refresh();
}

private void tableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    String getNIM = table.getValueAt(table.getSelectedRow(), 0).toString();
    String getNama = table.getValueAt(table.getSelectedRow(), 1).toString();
    // get data dari Table berdasarkan NIS
    try{
        String sql = "SELECT * FROM pendaftaran_kkp,mahasiswa,tempat_kkp "
            + "WHERE pendaftaran_kkp.id_mahasiswa=mahasiswa.id_mahasiswa AND "
            + "pendaftaran_kkp.id_tempat=tempat_kkp.id_tempat AND id_pendaftaran =
"+getNIM+"";
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(sql);
        while(rs.next()) {
            LIDPendaftaran.setText(rs.getString("id_pendaftaran"));
            LidMhs.setText(rs.getString("id_mahasiswa"));
            Lnim.setText(rs.getString("nim"));
            Lnama.setText(rs.getString("nama"));
            Ljurusan.setText(rs.getString("jurusan"));
            Lidtempatkkp.setText(rs.getString("id_tempat"));
            Lnamatempat.setText(rs.getString("nama_perusahaan"));
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

private void tNilaiKeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    String nilaiStr = tNilai.getText();
    int nilai = 0;

    try {
        nilai = Integer.parseInt(nilaiStr);
    } catch (NumberFormatException e) {
        LKet.setText("Nilai harus berupa angka.");
        return;
    }

    String keterangan;
    if (nilai >= 80 && nilai <= 100) {

```

```

        keterangan = "A (Sangat Baik)";
    } else if (nilai >= 70 && nilai < 80) {
        keterangan = "B (Baik)";
    } else if (nilai >= 60 && nilai < 70) {
        keterangan = "C (Sangat Cukup)";
    } else if (nilai >= 50 && nilai < 60) {
        keterangan = "D (Cukup)";
    } else if (nilai >= 0 && nilai < 50) {
        keterangan = "E (Ulang Tahun Depan)";
    } else {
        keterangan = "Nilai di luar jangkauan.";
    }

    LKet.setText(keterangan);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    dispose();
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JPrmkkp kkp = new JPrmkkp();
    kkp.setVisible(true);
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JPrmNilaikkp nkkp = new JPrmNilaikkp();
    nkkp.setVisible(true);
}

```

## **BAB IV**

### **PANDUAN PENGGUNAAN**

#### **4.1 View Halaman Utama Admin**

##### **a. Admin**

1. Admin memilih menu utilitas
2. Pilih sub menu login
3. lalu admin masukan pasword dan username
4. setelah itu klik login
5. jika data sesuai maka muncul ke halaman utama
6. di halaman utama admin terdapat 4 menu (Data master, Pendaftaran, Penilaian dan Penilaian Laporan)

##### **b. Data Master Dosen**

1. pilih menu data master
2. pilih sub menu data dosen
3. lalu jika sudah menampilkan form data,
4. admin input data dosen untuk memilih status kegiatan dosen, sebagai pembimbing kkn /kkp atau penilai atau tidak ada untuk memilih none
5. jika sudah simpan, jika data sudah lengkap maka langsung tersimpan,
6. kemudian jika ada data yang ingin di ubah bisa pilih data dosen, lalu input ubah data kemudian klik ubah,
7. jika ingin dihapus klik hapus, jika sudah semua data akan tampil kemudian jika sudah bisa klik close,
8. lalu kembali halaman utama.

##### **c. Data Mahasiswa**

1. Admin login lalu memilih sub menu data mahasiswa,
2. lalu input data mahasiswa siswa yang ikut kegiatan kkp atau kkn
3. jika sudah klik simpan,
4. jika ingin mengubah data mahasiswa bisa pilih data mahasiswa terlebih dahulu lalu input data yang di ubah,lalu klik ubah maka data akan berubah,
5. jika ingin dihapus pilih data terlebih dahulu lalu klik hapus,
6. jika sudah semua bisa klik close
7. lalu Kembali ke menu halaman utama

**d. Pengguna**

1. Admin login masuk lalu memilih sub menu data master pengguna untuk mengatur siapa saja yang ingin login ke dalam aplikasi,
2. jika sudah memilih bisa input data pengguna aplikasi sebagai mahasiswa, dosen atau admin ,
3. jika sudah input maka klik simpan, data pun akan tersimpan di tabel data,
4. jika ada data pengguna yang ingin di ubah bisa pilih data terlebih dahulu lalu pilih apa yang di ubah,
5. jika sudah klik ubah maka data akan ter ubah,
6. jika sudah bisa klik close aplikasi
7. dan Kembali halaman utama

**e. Tempat kkn**

1. Jika ingin menambah kan data kkn bisa pilih data master lalu klik data tempat kkn,
2. lalu input data ,
3. pilih dosen lalu klik simpan jika sudah lengkap maka akan tersimpan .
4. jika ada data pengguna yang ingin di ubah bisa pilih data terlebih dahulu lalu pilih apa yang di ubah,
5. jika sudah klik ubah maka data akan ter ubah,
6. jika sudah selesai semua bisa klik close aplikasi
7. dan Kembali halaman utama

**f. Tempat kkp**

1. Pilih menu data master ,
2. lalu klik sub menu data tempat kkp,
3. jika sudah terinput semua lalu klik simpan.
4. jika ada data tempat kkp yang ingin di ubah bisa pilih data terlebih dahulu lalu pilih apa yang di ubah,
5. jika sudah klik ubah maka data akan ter ubah,
6. jika sudah selesai semua bisa klik close aplikasi
7. dan Kembali halaman utama

**g. Laporan**

1. Admin pilih menu laporan
2. lalu pilih submenu laporan
3. jika sudah melihat laporan
4. bisa langsung klik close

## **4.2 Halaman Utama Dosen**

### a. Dosen

1. Login dahulu pilih utilitas
2. lalu,lalu masukan user name dan pasword dosen
3. setalah itu masuk ke halaman menu utama,
4. lalu pilih menu penilaian,
5. lalu pilih sub menu penilaian kkn/kkp,
6. pilih data mahasiswa terlebih dahulu ,
7. lalu input nilai mahasiswa
8. jika sudah klik submit,
9. jika sudah semua bisa klik close.s

## **4.3 Halaman Utama Mahasiswa**

### a. Tempat kkn

1. Pilih menu utili
2. lalu masukan username dan password yang sesuai,
3. jika sudah benar masuk menu utama,
4. kemudian pilih menu pendaftaran
5. kemudian pilih sub menu kkn/ kkp
6. dan pilih data mahasiswa
7. kemudian pilih tempat kkn jika setuju klik setuju,
8. kemudian klik daftar,
9. jika data sudah lengkap data tersimpan
10. kemudian jika sudah semua klik close
11. Kembali ke halaman Utama