

## **FINAL PROJECT UAS**

**Pemrograman Berorientasi Objek**

**“Sistem Perpustakaan Kampus”**



Dosen Pengampu : Taufik Ridwan, M.T.

Disusun oleh :

Kelompok 2

Fadilah Nur Lutfah	2310631250051
Fauzan Farhan Gayo	2310631250056
Hafshah	2310631250020
Sri Novellaputri Ramadhany	2310631250078

**PROGRAM STUDI SISTEM INFORMASI**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS SINGAPERBANGSA KARAWANG**

**2025**

## KATA PENGANTAR

Puji syukur kehadirat Allah Subhanahu Wata'ala yang Maha Pengasih lagi Maha Penyayang yang telah memberikan rahmat dan karunia-Nya sehingga kami dapat menyelesaikan Final Project UAS untuk Mata Kuliah Pemrograman Berorientasi Objek yang membahas tentang "Sistem Perpustakaan Kampus" dapat kami selesaikan dengan lancar. Shalawat serta salam kami curahkan kepada baginda Rasulullah Muhammad Shallallahu Alaihi Wasallam beserta keluarga dan para sahabatnya. Laporan ini kami susun dengan maksimal dan mendapatkan bantuan dari berbagai pihak sehingga dapat memperlancar pembuatan laporan ini.

Untuk itu kami menyampaikan banyak terima kasih kepada:

1. Rektor Universitas Singaperbangsa Karawang, Prof. Dr. Ade Maman Suherman, S.H., M.Sc.
2. Dosen Mata Kuliah Pemrograman Berorientasi Objek, Bapak Taufik Ridwan, M.T., yang telah memberikan kami ilmu yang bermanfaat.
3. Orang tua dan teman-teman yang telah memberi dukungan.

Kami juga sampaikan, jika di dalam laporan ini terdapat hal yang tidak sesuai dengan harapan, kami dengan senang hati menerima masukan, kritikan dan saran dari pembaca. Laporan ini masih memiliki kekurangan dan tidak sempurna. Akhir kata kami berharap semoga laporan yang kami hasilkan dapat menjadi bermanfaat kepada sesama.

Karawang, 28 Mei 2025

Kelompok 2

## DAFTAR ISI

<b>KATA PENGANTAR.....</b>	<b>1</b>
<b>DAFTAR ISI.....</b>	<b>2</b>
<b>BAB I</b>	
<b>PENDAHULUAN.....</b>	<b>3</b>
1.1 Latar Belakang.....	3
1.2 Rumusan Masalah.....	3
1.3 Tujuan.....	3
<b>BAB II</b>	
<b>PEMBAHASAN.....</b>	<b>5</b>
2.1 Fitur-Fitur dari Aplikasi yang Dibuat.....	5
2.2 Konsep OOP dari Kode Program.....	5
1. Encapsulation (Enkapsulasi).....	6
2. Inheritance (Pewarisan).....	6
3. Polymorphism (Polimorfisme).....	7
2.3 Perancangan menggunakan Unified Modelling Language.....	8
2.4 Implementasi Kode Program.....	17
2.5 Pengujian.....	37
<b>BAB III</b>	
<b>PENUTUP.....</b>	<b>42</b>
3.1 Kesimpulan.....	42

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Perpustakaan merupakan salah satu fasilitas yang sangat penting di lingkungan kampus yang menyediakan berbagai referensi untuk mendukung kegiatan akademik mahasiswa. Dalam pengelolaan perpustakaan mencakup berbagai kegiatan seperti pencatatan data buku, proses peminjaman dan pengembalian buku, serta manajemen data anggota. Sayangnya, banyak perpustakaan yang masih menggunakan sistem manual atau semi-digital yang kurang efisien dan rentan terhadap kesalahan.

Proses pencatatan yang dilakukan secara manual memiliki beberapa kelemahan, diantaranya seperti resiko kehilangan data, duplikasi informasi, dan keterbatasan dalam pencarian maupun pelacakan status buku. Di sisi lain, kebutuhan akan layanan yang cepat, akurat, dan mudah diakses semakin meningkat seiring dengan perkembangan teknologi informasi dan ekspektasi pengguna terhadap layanan perpustakaan yang modern.

Melihat permasalahan tersebut, dibutuhkan solusi berupa sistem informasi perpustakaan yang terkomputerisasi dan mampu mempermudah dalam pengelolaan data. Melalui proyek pengembangan sistem perpustakaan ini, mahasiswa tidak hanya memahami penerapan konsep OOP secara praktis, tetapi juga mampu menghasilkan solusi yang bermanfaat dan aplikatif dalam kehidupan nyata. Proyek ini diharapkan dapat menjadi langkah awal dalam mengembangkan sistem informasi yang lebih kompleks dan profesional di masa mendatang.

### 1.2 Rumusan Masalah

1. Bagaimana merancang dan mengimplementasikan sistem perpustakaan berbasis Java dengan konsep *Object Oriented Programming* (OOP) yang dapat mempermudah pengelolaan data buku, anggota, serta aktivitas peminjaman dan pengembalian?
2. Bagaimana sistem dapat membedakan peran antara pengunjung dan pustakawan dalam pengelolaan dan pemantauan aktivitas perpustakaan?
3. Bagaimana menyediakan fitur peminjaman buku secara online bagi pengunjung secara efisien dan akurat?
4. Bagaimana merancang antarmuka pengguna (GUI) yang intuitif dan user-friendly untuk kedua role pengguna?
5. Bagaimana sistem dapat meminimalisir kesalahan pencatatan dan meningkatkan efisiensi kerja pustakawan melalui digitalisasi proses?

### 1.3 Tujuan

1. Mengembangkan sistem perpustakaan digital berbasis Java dengan penerapan konsep OOP sebagai solusi atas pengelolaan data perpustakaan yang masih manual.
2. Membangun sistem dengan dua role pengguna, yaitu pengunjung dan pustakawan, dengan hak akses dan fitur yang sesuai perannya masing-masing.

3. Menyediakan fitur peminjaman dan pengembalian buku secara digital yang dapat diakses oleh pengunjung secara online.
4. Meningkatkan efisiensi pustakawan dalam mengelola data anggota, buku, serta aktivitas perpustakaan melalui dashboard monitoring.
5. Merancang antarmuka grafis yang memudahkan interaksi pengguna dengan sistem sehingga meningkatkan pengalaman pengguna (*user experience*).
6. Memberikan pengalaman praktis dalam pengembangan sistem informasi menggunakan pendekatan OOP serta memperkuat pemahaman terhadap struktur dan logika pemrograman berorientasi objek.

## **BAB II**

### **PEMBAHASAN**

#### **2.1 Fitur-Fitur dari Aplikasi yang Dibuat**

##### **A. Pengunjung**

1. Melihat Daftar Buku: Fitur ini memungkinkan pengunjung melihat seluruh koleksi buku yang tersedia di perpustakaan kampus. Informasi yang ditampilkan mencakup id buku, judul, penulis, penerbit, tahun terbit, dan status ketersediaan buku.
2. Peminjaman Buku: Pengunjung dapat melakukan peminjaman buku secara online melalui sistem perpustakaan. Fitur ini menampilkan daftar buku yang bisa dipinjam dengan batas maksimal peminjaman sesuai kebijakan kampus.
3. Riwayat Peminjaman: Fitur ini menampilkan riwayat seluruh peminjaman buku oleh pengunjung, termasuk informasi tanggal pinjam, tanggal pengembalian, status pengembalian, dan denda jika ada keterlambatan.

##### **B. Pustakawan**

1. Kelola Buku: Fitur ini digunakan untuk menambahkan, mengedit, atau menghapus data buku yang tersedia di perpustakaan. Pustakawan dapat memperbarui informasi buku seperti judul, penulis, penerbit, tahun terbit, dan status ketersediaan buku.
2. Kelola Anggota: Pustakawan dapat mengelola data anggota perpustakaan (mahasiswa & dosen). Termasuk pendaftaran anggota baru, edit data anggota, dan penghapusan anggota yang sudah tidak aktif.
3. Kelola Peminjaman: Fitur ini digunakan untuk mencatat transaksi peminjaman buku oleh anggota, baik secara manual maupun verifikasi dari sistem peminjaman online. Termasuk pengecekan status buku dan batas waktu peminjaman.
4. Proses Pengembalian: Fitur ini digunakan oleh pustakawan untuk mencatat bahwa seorang anggota (pengunjung) telah mengembalikan buku secara langsung. Proses pengembalian tidak otomatis, melainkan dilakukan setelah pustakawan menerima fisik buku dari peminjam, lalu menandai status pengembalian di sistem.
5. Denda: Fitur ini mencatat dan menampilkan daftar denda yang dikenakan pada anggota karena keterlambatan pengembalian. Termasuk rincian jumlah denda, status pembayaran, dan histori keterlambatan.

#### **2.2 Konsep OOP dari Kode Program**

Pada pengembangan proyek ini, paradigma Pemrograman Berbasis Objek (*Object Oriented Programming/OOP*) digunakan sebagai pendekatan utama dalam menyusun struktur dan logika program. OOP merupakan paradigma yang memodelkan komponen perangkat lunak sebagai objek-objek yang saling berinteraksi. Objek-objek ini dibentuk berdasarkan

kelas (*class*), yang mendeskripsikan atribut (*properties*) dan perilaku (*methods*) dari suatu entitas dalam sistem.

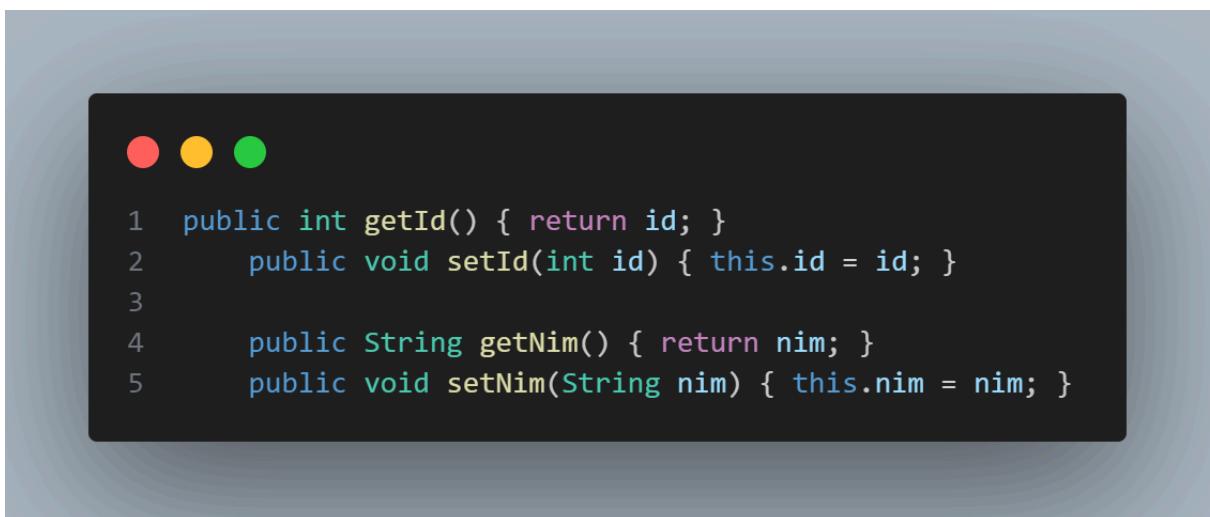
Pendekatan OOP memberikan berbagai keuntungan, antara lain struktur program yang lebih terorganisir, kemudahan dalam pemeliharaan kode (*Maintainability*), kemampuan untuk mengembangkan fitur baru dengan cepat (*Scalability*), serta penerapan prinsip *DRY (Don't Repeat Yourself)* melalui penggunaan kembali kode (*Code reuse*). Dalam proyek ini, penerapan konsep OOP diwujudkan melalui:

### 1. *Encapsulation* (Enkapsulasi)

Enkapsulasi adalah mekanisme yang memungkinkan data (atribut) dan perilaku (metode) dibungkus ke dalam satu kesatuan unit bernama kelas. Akses terhadap data dikendalikan melalui modifier seperti *private*, *protected*, dan *public*, sehingga data tidak bisa diakses atau diubah secara langsung dari luar kelas.

Dalam program ini, setiap entitas seperti pengguna, buku, atau transaksi peminjaman direpresentasikan sebagai kelas yang memiliki atribut privat. Misalnya, kelas Buku memiliki atribut seperti judul, penulis, dan status, yang hanya dapat diakses melalui metode *getter* dan *setter*. Dengan demikian, enkapsulasi menjaga integritas data, meningkatkan keamanan, serta memudahkan proses debugging dan validasi logika.

Contoh kode program dalam sistem:

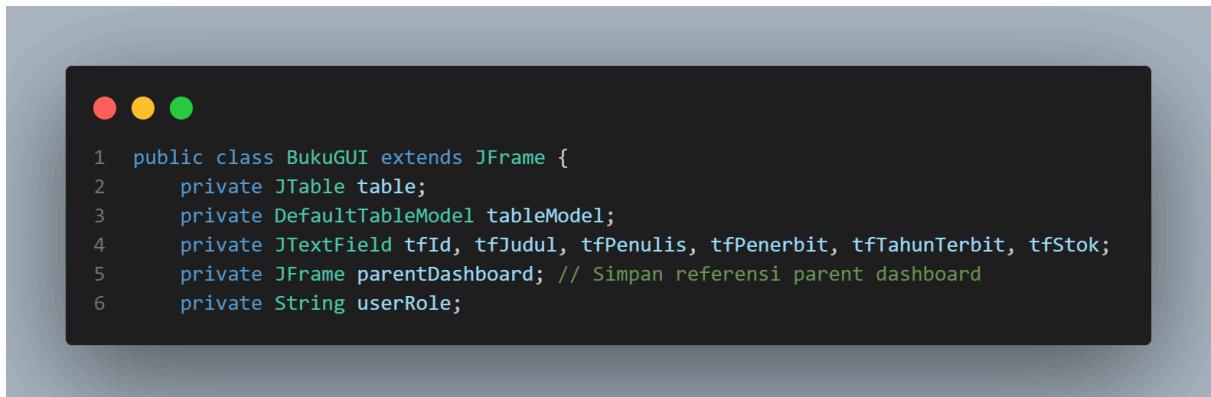


### 2. *Inheritance* (Pewarisan)

*Inheritance* adalah konsep di mana sebuah kelas (*subclass*) dapat mewarisi properti dan metode dari kelas lain (*superclass*). Ini memungkinkan developer untuk membangun hierarki kelas dan menghindari duplikasi kode.

Dalam proyek ini, *inheritance* diterapkan untuk membedakan peran dalam sistem, misalnya kelas Pengguna sebagai superclass yang kemudian diturunkan menjadi kelas Pengunjung dan Pustakawan. Kelas turunan dapat menambahkan atau memodifikasi perilaku dari kelas induknya, tanpa harus menulis ulang semua logika yang sama. Dengan pewarisan, struktur program menjadi lebih modular dan mendukung prinsip *code reuse* secara optimal.

Contoh kode program dalam sistem:



```
● ● ●
1 public class BukuGUI extends JFrame {
2     private JTable table;
3     private DefaultTableModel tableModel;
4     private JTextField tfId, tfJudul, tfPenulis, tfPenerbit, tfTahunTerbit, tfStok;
5     private JFrame parentDashboard; // Simpan referensi parent dashboard
6     private String userRole;
```

### 3. *Polymorphism* (Polimorfisme)

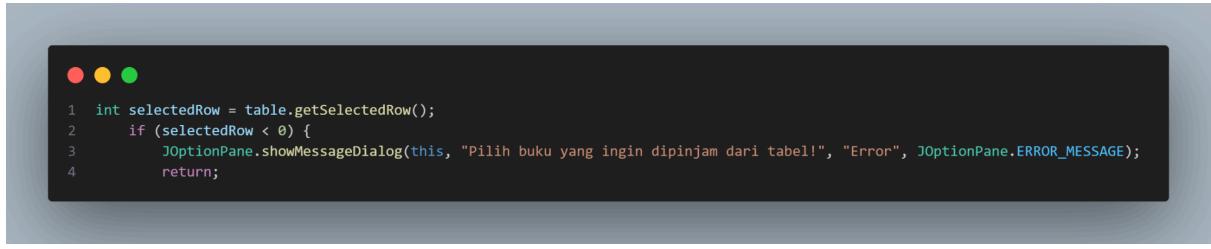
Polimorfisme memungkinkan objek untuk diperlakukan sebagai bentuk umum dari superclass-nya, namun berperilaku sesuai implementasi spesifik pada subclass. Polimorfisme terbagi menjadi dua: compile-time polymorphism (*method overloading*) dan runtime polymorphism (*method overriding*).

Dalam program ini, diterapkan *runtime polymorphism*, contohnya metode tampilkanInfo() yang didefinisikan dalam kelas Pengguna, kemudian di override dalam kelas Pustakawan dan Pengunjung untuk menampilkan informasi yang sesuai dengan peran masing-masing. Dengan cara ini, program dapat memanggil metode dengan cara yang sama namun menghasilkan perilaku yang berbeda tergantung objek yang digunakan.

Contoh kode program dalam sistem:



```
● ● ●
1 String nim = JOptionPane.showInputDialog(this, "Masukkan NIM/NIP Anda:");
2     if (nim == null || nim.trim().isEmpty()) {
3         JOptionPane.showMessageDialog(this, "NIM/NIP wajib diisi!", "Error", JOptionPane.ERROR_MESSAGE);
4         return;
5     }
```

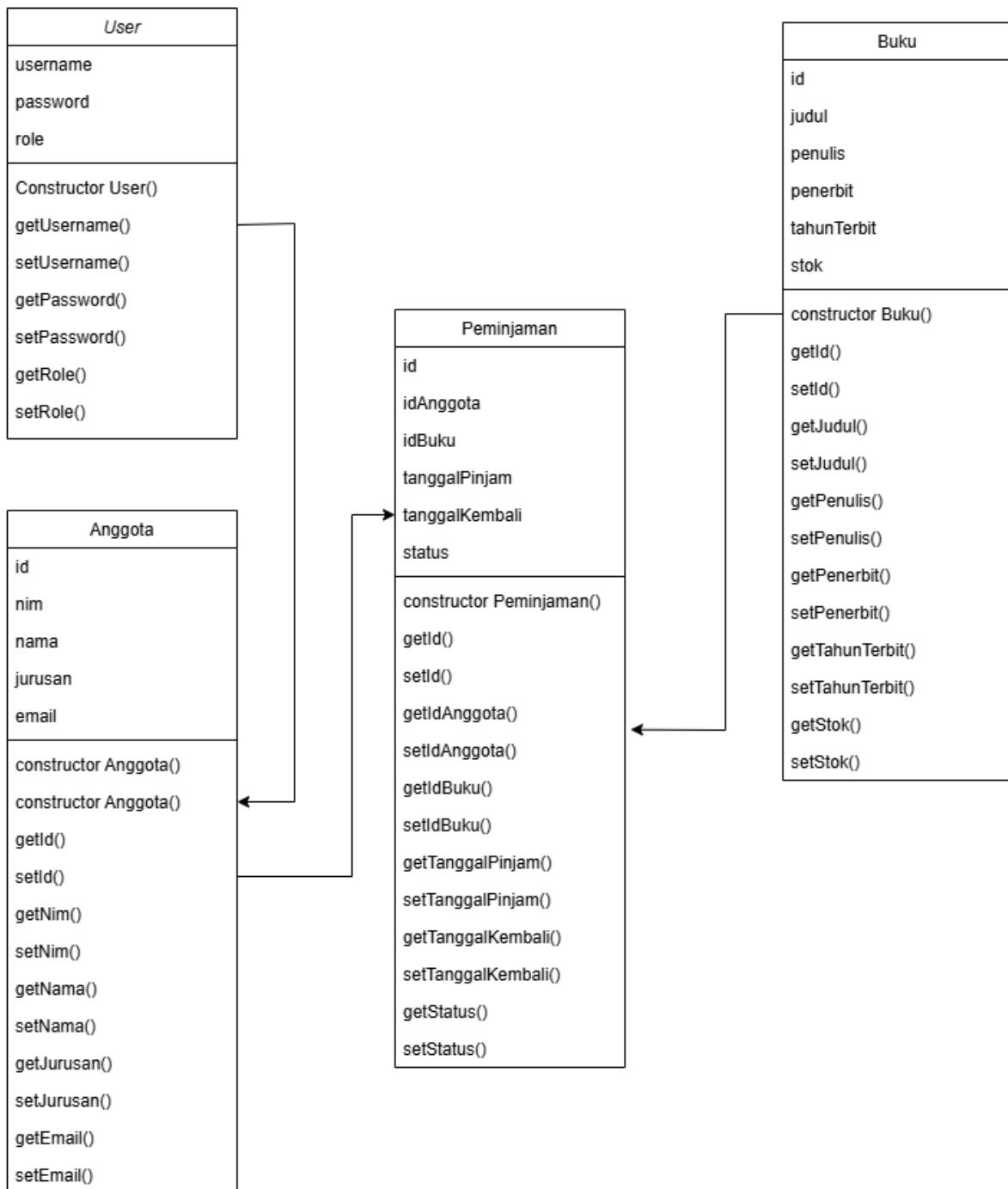


```
1 int selectedRow = table.getSelectedRow();
2 if (selectedRow < 0) {
3     JOptionPane.showMessageDialog(this, "Pilih buku yang ingin dipinjam dari tabel!", "Error", JOptionPane.ERROR_MESSAGE);
4     return;
```

Secara keseluruhan, penerapan prinsip-prinsip OOP dalam proyek ini berperan penting dalam membentuk sistem yang bersifat modular, fleksibel, dan mudah dikembangkan. Setiap kelas dirancang untuk memiliki tanggung jawab spesifik (*single responsibility*), serta saling berinteraksi melalui mekanisme pewarisan dan komposisi. Dengan pendekatan ini, sistem menjadi lebih mudah untuk diuji, diperluas, dan dipelihara seiring waktu.

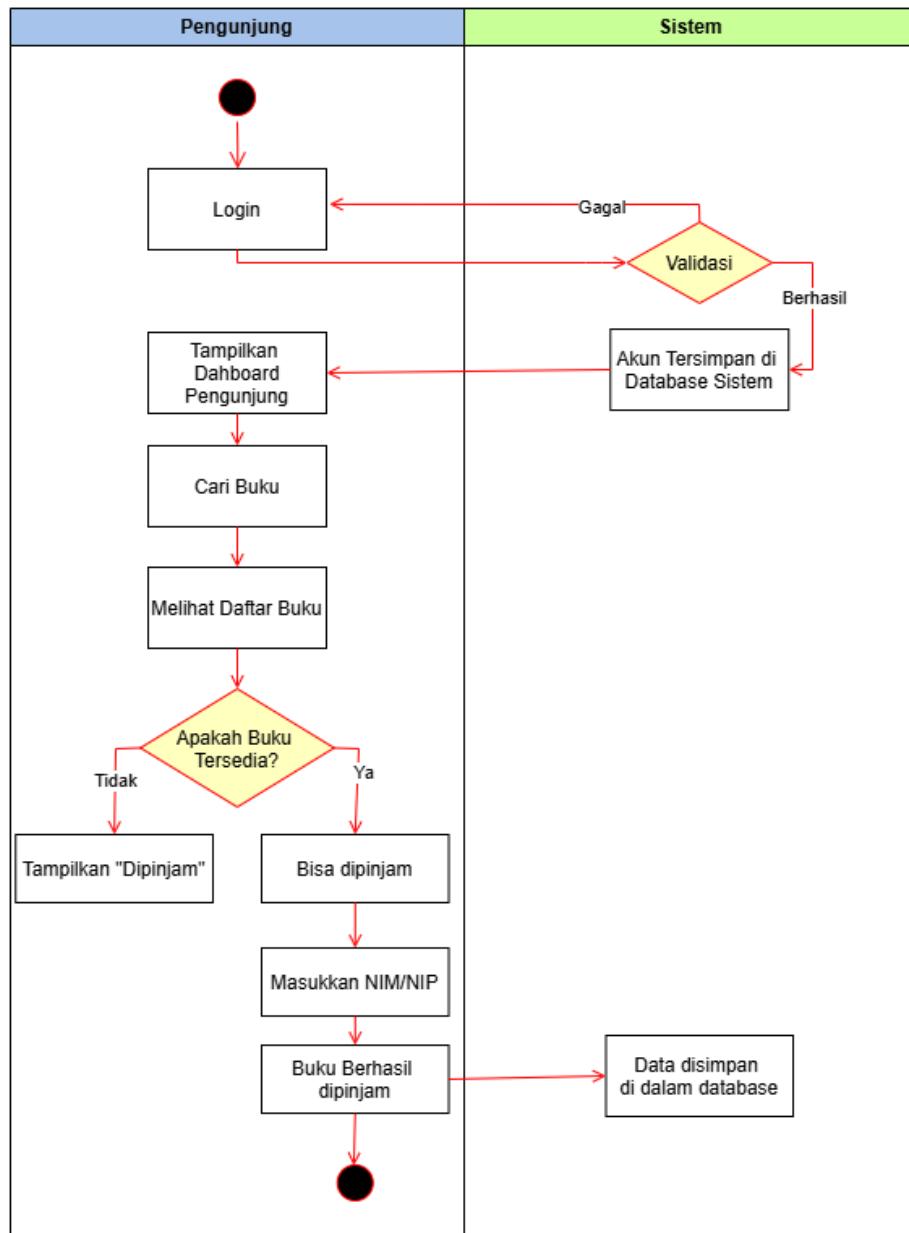
### 2.3 Perancangan menggunakan Unified Modelling Language

#### 1. Class Diagram



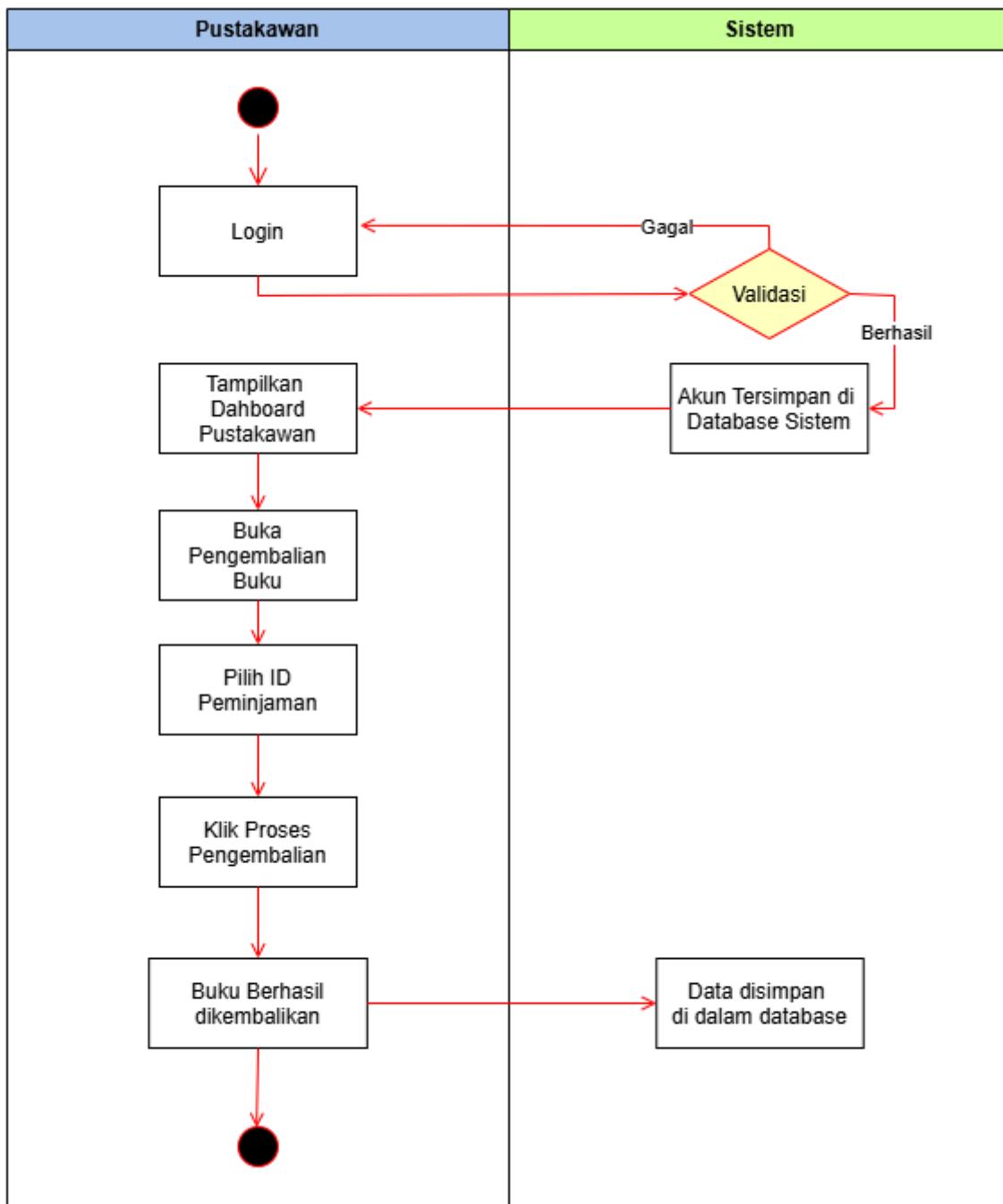
## 2. Activity Diagram

### A. Pengunjung

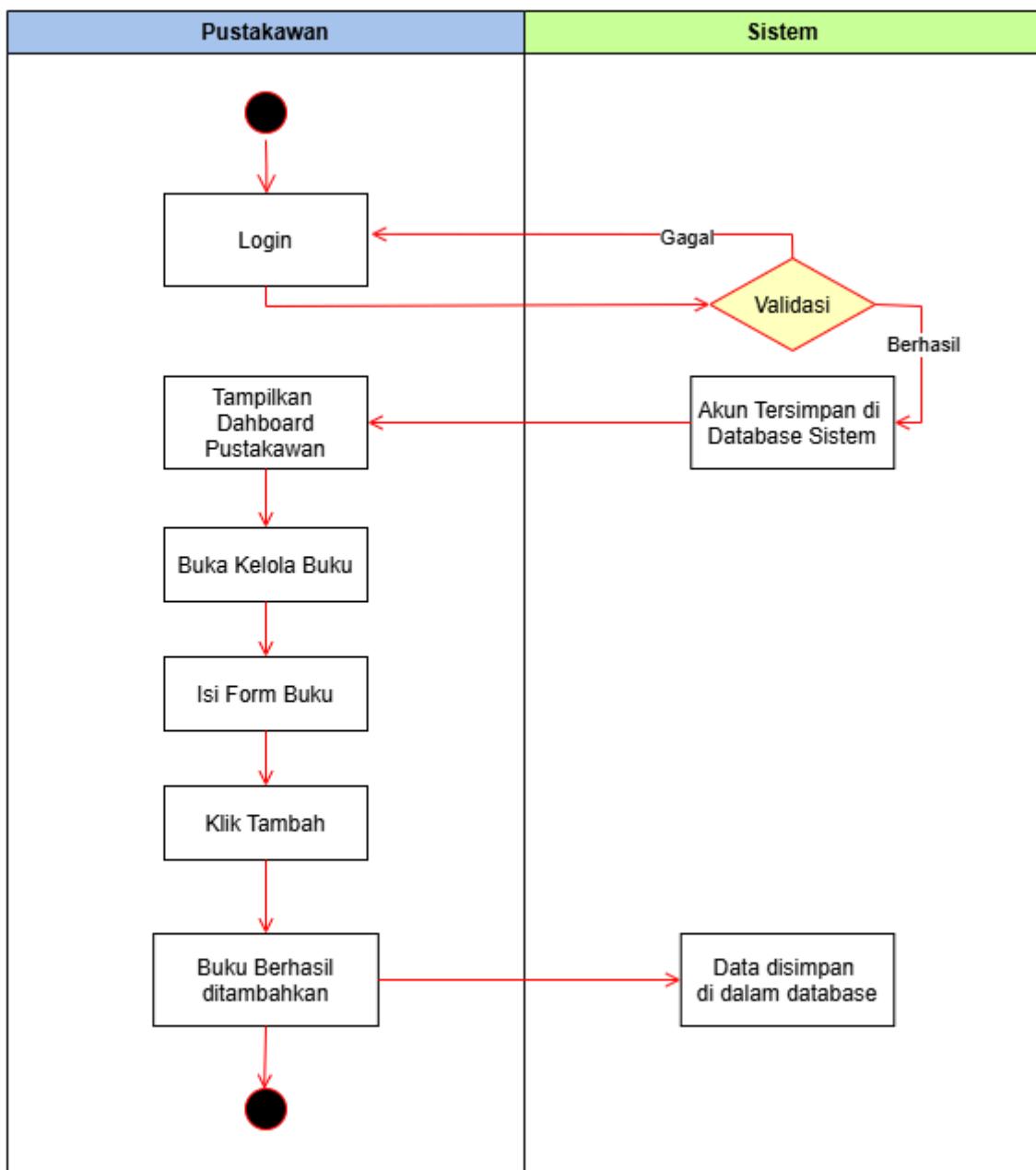


## B. Pustakawan

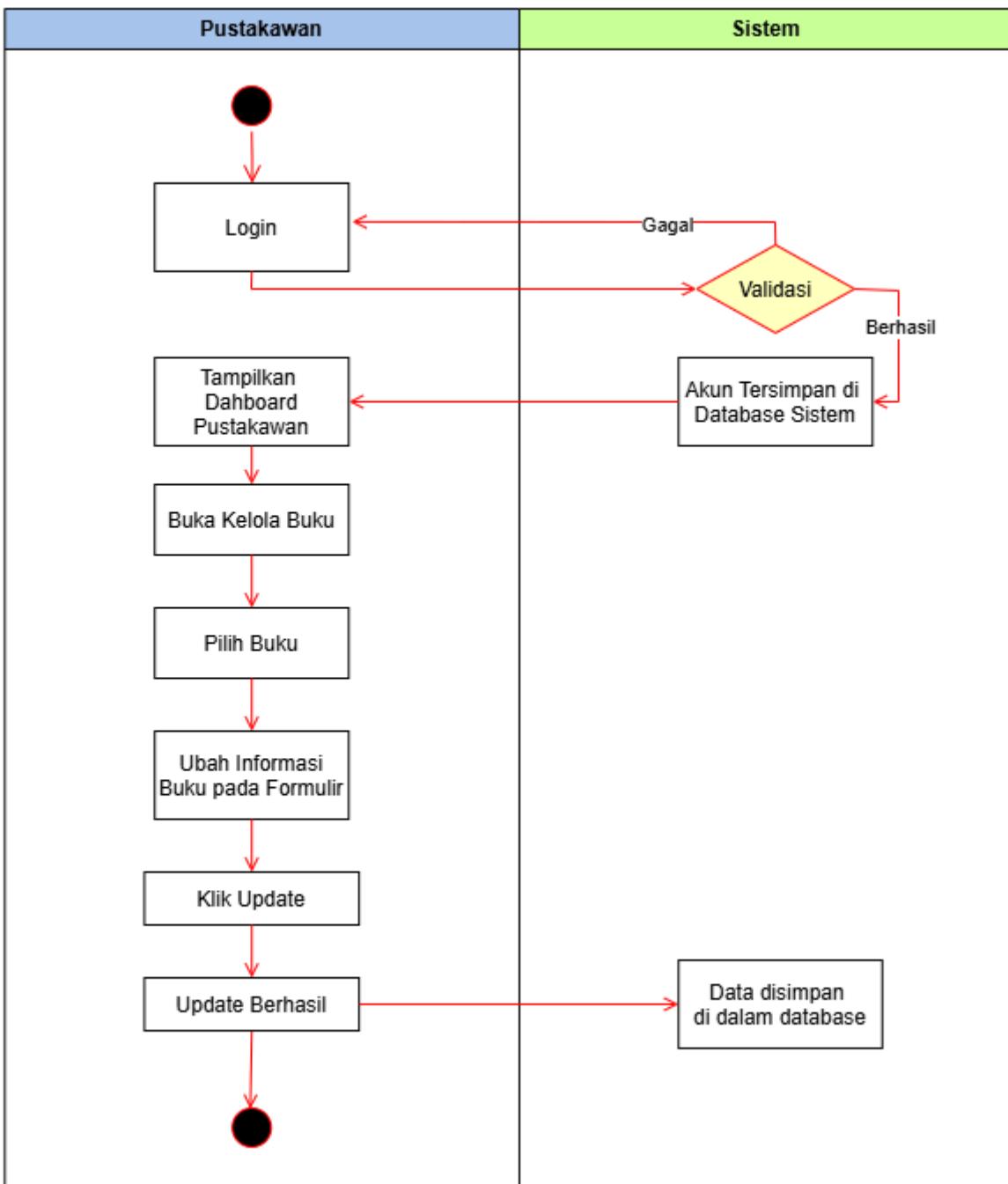
- Alur Pengembalian Buku



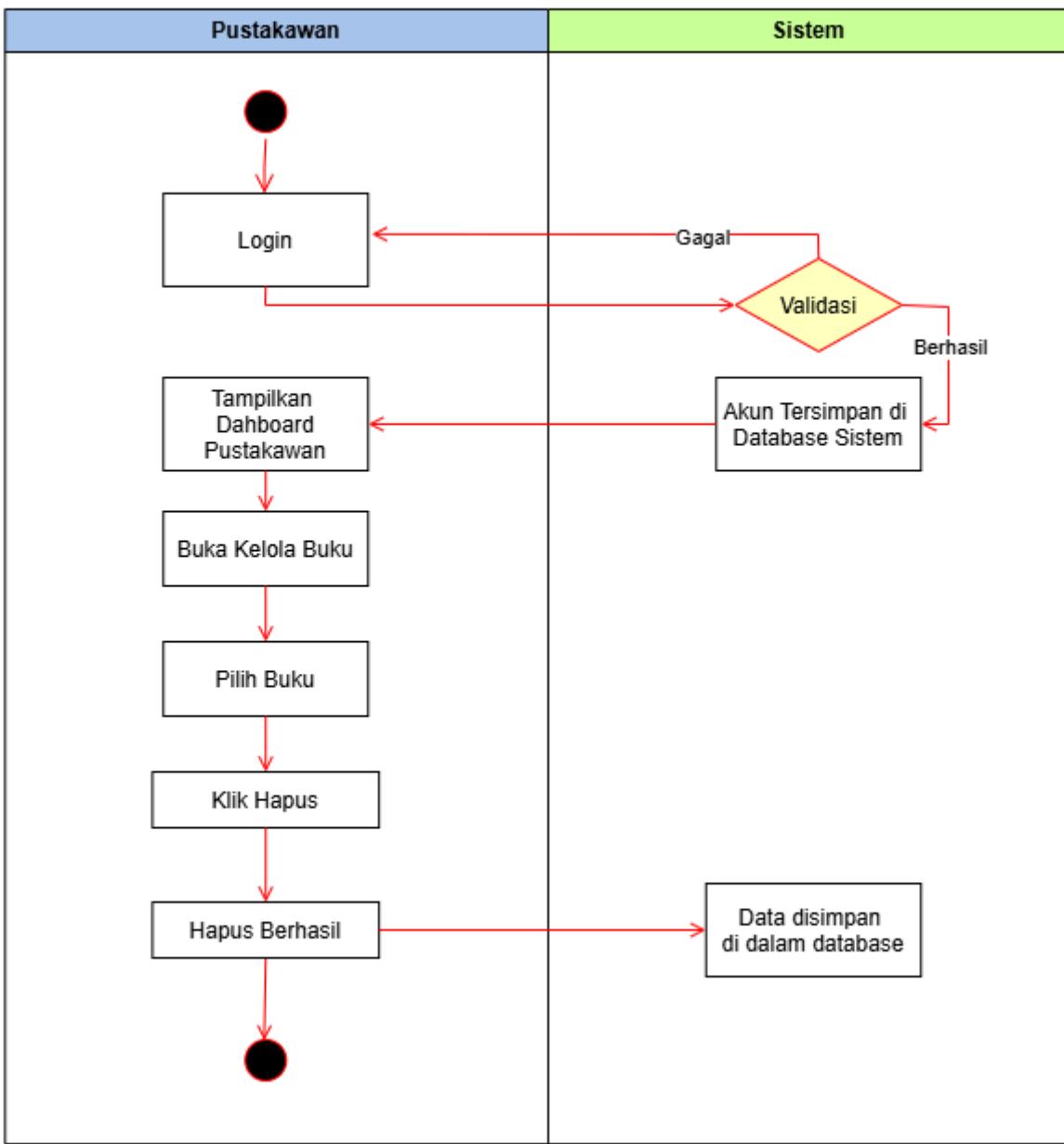
- Alur Tambah Buku



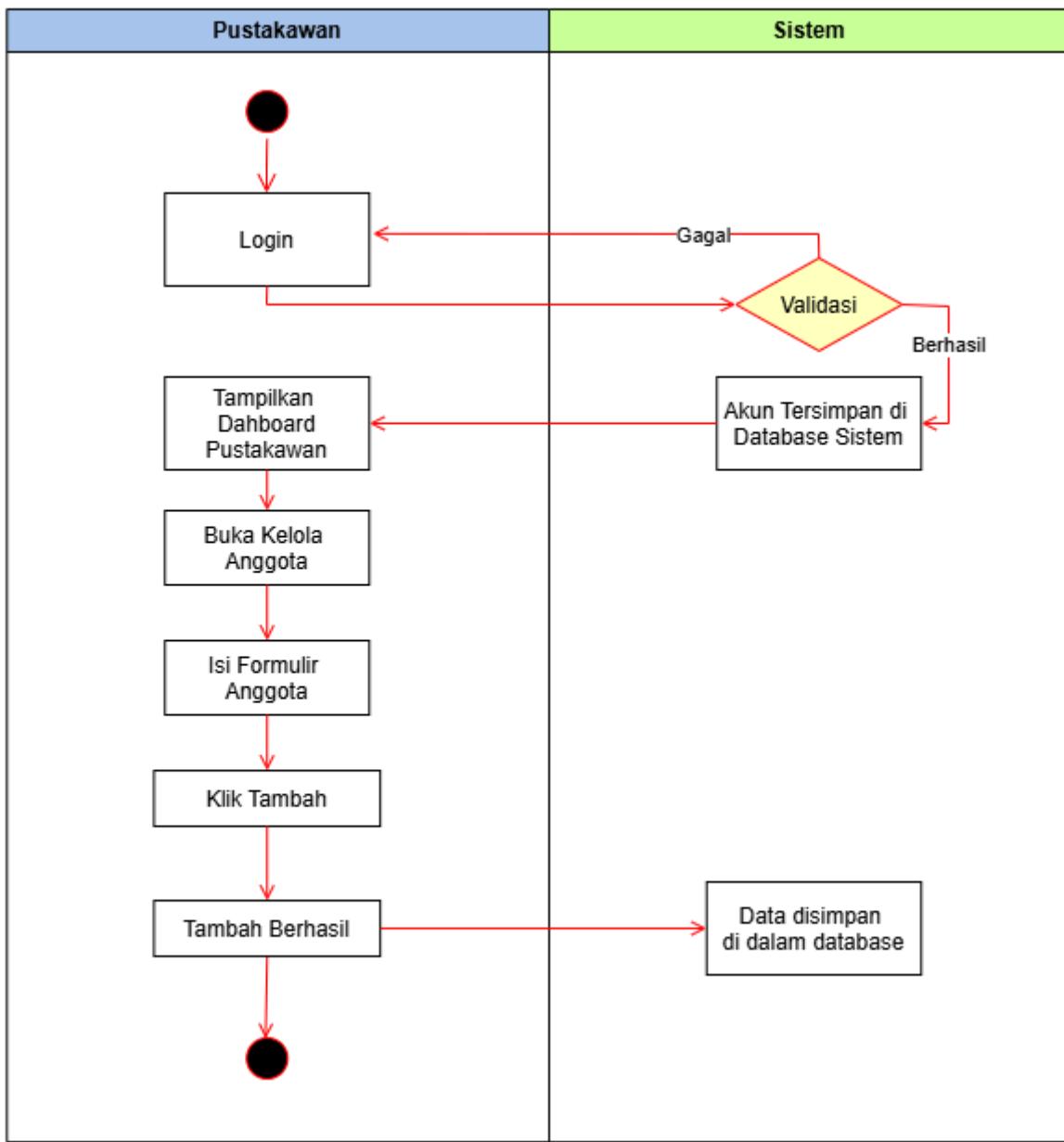
- Alur Update Buku



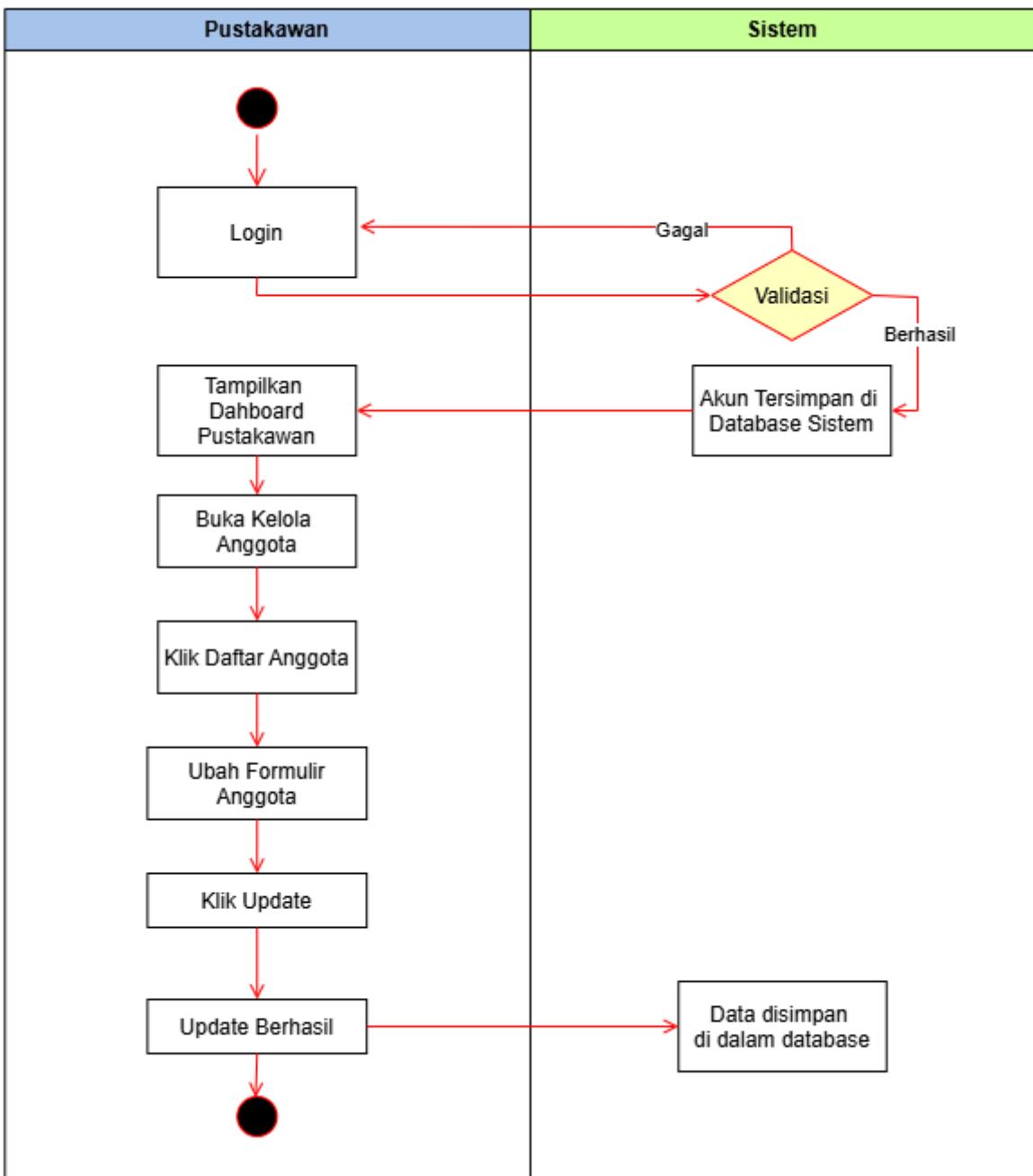
- Alur Hapus Buku



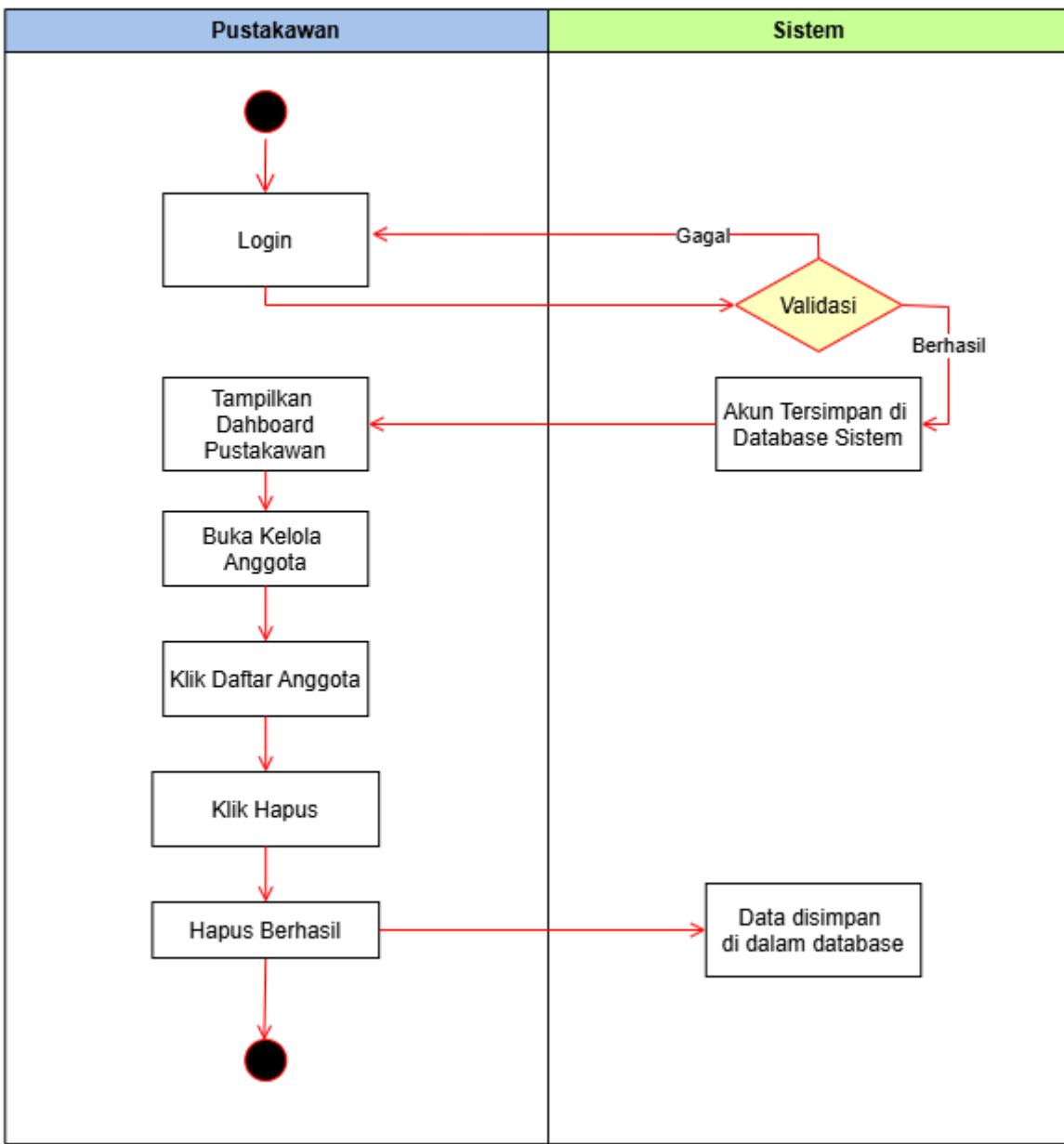
- Alur Tambah Anggota



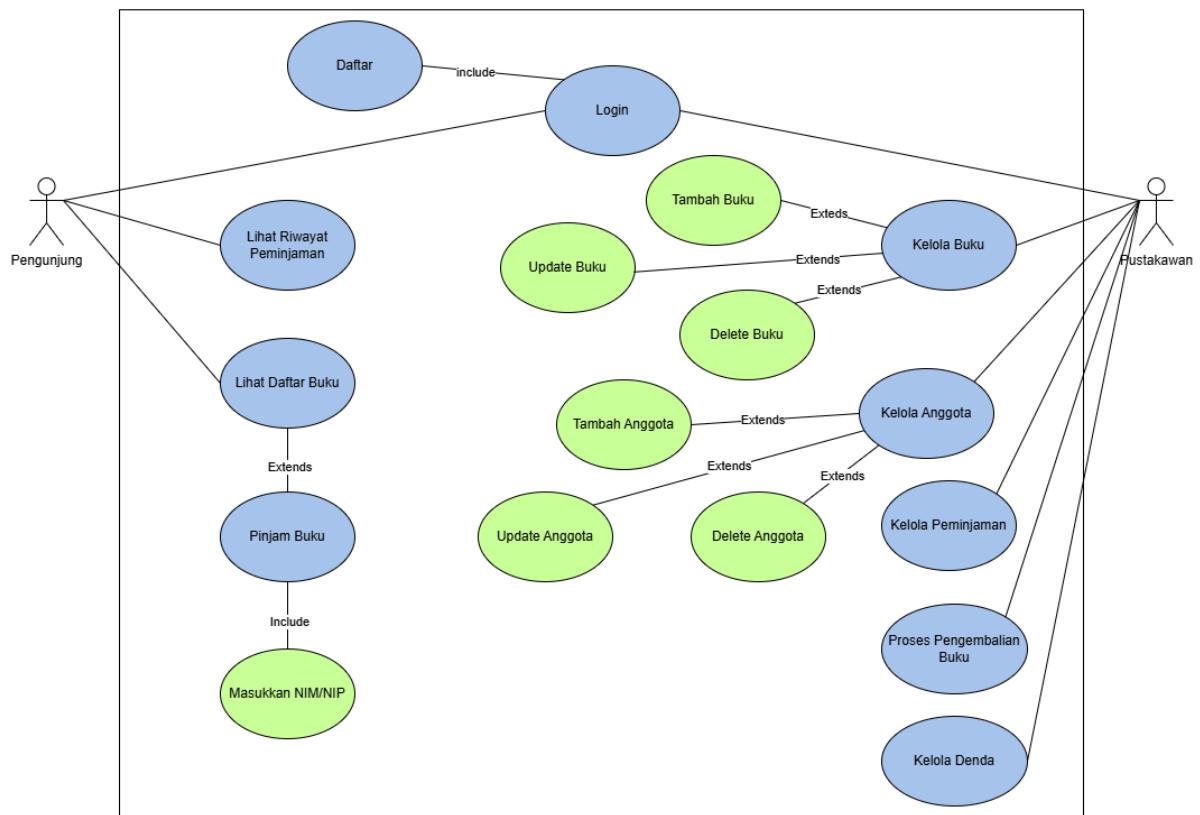
- Alur Update Anggota



- Alur Hapus Anggota



### 3. Use Case Diagram



### 2.4 Implementasi Kode Program

- **Main.java**

```
● Main.java

1 import view.LoginGUI;
2
3 import javax.swing.*;
4
5 public class Main {
6     public static void main(String[] args) {
7         // Tampilan modern look & feel
8         try {
9             UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
10        } catch (Exception ignored) {}
11
12        SwingUtilities.invokeLater(() -> new LoginGUI());
13    }
14 }
15
```

- **DatabaseConnection.java**

```
1 package db;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class DatabaseConnection {
8     private static final String URL = "jdbc:mysql://localhost:3306/perpus";
9     private static final String USER = "root";
10    private static final String PASSWORD = "";
11
12    public static Connection getConnection() {
13        try {
14            Class.forName("com.mysql.cj.jdbc.Driver");
15            return DriverManager.getConnection(URL, USER, PASSWORD);
16        } catch (ClassNotFoundException | SQLException e) {
17            System.out.println("[DB ERROR] " + e.getMessage());
18            return null;
19        }
20    }
21 }
22
```

- **AnggotaDAO.java**



```

1 package dao;
2
3 import db.DatabaseConnection;
4 import model.Anggota;
5
6 import java.sql.*;
7 import java.util.ArrayList;
8 import java.util.List;
9
10 public class AnggotaDAO {
11     public List<Anggota> getAll() {
12         List<Anggota> list = new ArrayList<>();
13         try (Connection conn = DatabaseConnection.getConnection()) {
14             String sql = "SELECT * FROM anggota";
15             PreparedStatement stmt = conn.prepareStatement(sql);
16             ResultSet rs = stmt.executeQuery();
17             while (rs.next()) {
18                 Anggota a = new Anggota(
19                     rs.getInt("id"),
20                     rs.getString("nim"),
21                     rs.getString("nama"),
22                     rs.getString("jurusan"),
23                     rs.getString("email")
24                 );
25                 list.add(a);
26             }
27         } catch (SQLException e) {
28             e.printStackTrace();
29         }
30         return list;
31     }
32
33     public static Anggota login(String username, String password) {
34         Anggota anggota = null;
35         try (Connection conn = DatabaseConnection.getConnection()) {
36             String sql = "SELECT * FROM anggota WHERE username = ? AND password = ?";
37             PreparedStatement stmt = conn.prepareStatement(sql);
38             stmt.setString(1, username);
39             stmt.setString(2, password);
40             ResultSet rs = stmt.executeQuery();
41             if (rs.next()) {
42                 anggota = new Anggota(
43                     rs.getInt("id"),
44                     rs.getString("nim"),
45                     rs.getString("nama"),
46                     rs.getString("jurusan"),
47                     rs.getString("email")
48                 );
49             }
50         } catch (SQLException e) {
51             e.printStackTrace();
52         }
53         return anggota;
54     }
55
56     public static void tambahAnggota(Anggota a) {
57         String sql = "INSERT INTO anggota (nim, nama, jurusan, email) VALUES (?, ?, ?, ?)";
58         try (Connection conn = DatabaseConnection.getConnection();
59              PreparedStatement stmt = conn.prepareStatement(sql)) {
56
59             stmt.setString(1, a.getNim());
60             stmt.setString(2, a.getName());
61             stmt.setString(3, a.getJurusan());
62             stmt.setString(4, a.getEmail());
63             stmt.executeUpdate();
64
65         } catch (SQLException e) {
66             e.printStackTrace();
67         }
68     }
69
70     public static void updateAnggota(Anggota a) {
71         String sql = "UPDATE anggota SET nama = ?, jurusan = ?, email = ? WHERE nim = ?";
72         try (Connection conn = DatabaseConnection.getConnection();
73              PreparedStatement stmt = conn.prepareStatement(sql)) {
74
75             stmt.setString(1, a.getName());
76             stmt.setString(2, a.getJurusan());
77             stmt.setString(3, a.getEmail());
78             stmt.setString(4, a.getNim());
79             stmt.executeUpdate();
80
81         } catch (SQLException e) {
82             e.printStackTrace();
83         }
84     }
85
86     public static void hapusAnggota(String nim) {
87         String sql = "DELETE FROM anggota WHERE nim = ?";
88         try (Connection conn = DatabaseConnection.getConnection();
89              PreparedStatement stmt = conn.prepareStatement(sql)) {
90
91             stmt.setString(1, nim);
92             stmt.executeUpdate();
93
94         } catch (SQLException e) {
95             e.printStackTrace();
96         }
97     }
98 }
99 }
```

## ● BukuDAO.java

```
1 package dao;
2
3 import model.Anggota;
4 import model.Buku;
5 import db.DatabaseConnection;
6
7 import java.sql.*;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 public class BukuDAO {
12     public static void simpan(Buku buku) {
13         String sql = "INSERT INTO buku (judul, penulis, penerbit, tahun_terbit, stok) VALUES (?, ?, ?, ?, ?)";
14         try (Connection conn = DatabaseConnection.getConnection();
15              PreparedStatement stmt = conn.prepareStatement(sql)) {
16             stmt.setString(1, buku.getJudul());
17             stmt.setString(2, buku.getPenulis());
18             stmt.setString(3, buku.getPenerbit());
19             stmt.setInt(4, buku.getTahunTerbit());
20             stmt.setInt(5, buku.getStok());
21             stmt.executeUpdate();
22         } catch (SQLException e) {
23             e.printStackTrace();
24         }
25     }
26
27     public static List<Buku> getAll() {
28         List<Buku> list = new ArrayList<>();
29         String sql = "SELECT * FROM buku";
30         try (Connection conn = DatabaseConnection.getConnection();
31              PreparedStatement stmt = conn.prepareStatement(sql);
32              ResultSet rs = stmt.executeQuery()) {
33             while (rs.next()) {
34                 Buku buku = new Buku(
35                     rs.getString("id"),
36                     rs.getString("judul"),
37                     rs.getString("penulis"),
38                     rs.getString("penerbit"),
39                     rs.getInt("tahun_terbit"),
40                     rs.getInt("stok")
41                 );
42                 list.add(buku);
43             }
44         } catch (SQLException e) {
45             e.printStackTrace();
46         }
47         return list;
48     }
49
50     public static void hapusBuku(String id) {
51         String sql = "DELETE FROM buku WHERE id = ?";
52         try (Connection conn = DatabaseConnection.getConnection();
53              PreparedStatement stmt = conn.prepareStatement(sql)) {
54             stmt.setString(1, id);
55             stmt.executeUpdate();
56         } catch (SQLException e) {
57             e.printStackTrace();
58         }
59     }
60
61     public static void tambahBuku(Buku a) {
62         String sql = "INSERT INTO buku (id, judul, penulis, penerbit, stok) VALUES (?, ?, ?, ?, ?, ?)";
63         try (Connection conn = DatabaseConnection.getConnection();
64              PreparedStatement stmt = conn.prepareStatement(sql)) {
65             stmt.setString(1, a.getId());
66             stmt.setString(2, a.getJudul());
67             stmt.setString(3, a.getPenulis());
68             stmt.setString(4, a.getPenerbit());
69             stmt.setInt(5, a.getTahunTerbit());
70             stmt.setInt(6, a.getStok());
71             stmt.executeUpdate();
72         } catch (SQLException e) {
73             e.printStackTrace();
74         }
75     }
76
77     public static void updateBuku(Buku a) {
78         String sql = "UPDATE buku SET judul = ?, penulis = ?, penerbit = ?, tahun_terbit = ?, stok = ? WHERE id = ?";
79         try (Connection conn = DatabaseConnection.getConnection();
80              PreparedStatement stmt = conn.prepareStatement(sql)) {
81             stmt.setString(1, a.getJudul());
82             stmt.setString(2, a.getPenulis());
83             stmt.setString(3, a.getPenerbit());
84             stmt.setInt(4, a.getTahunTerbit());
85             stmt.setInt(5, a.getStok());
86             stmt.setString(6, a.getId());
87             stmt.executeUpdate();
88         } catch (SQLException e) {
89             e.printStackTrace();
90         }
91     }
92
93     public static void updateStok(String idBuku, int stokBaru) {
94         String sql = "UPDATE buku SET stok = ? WHERE id = ?";
95         try (Connection conn = DatabaseConnection.getConnection();
96              PreparedStatement stat = conn.prepareStatement(sql)) {
97             stat.setInt(1, stokBaru);
98             stat.setString(2, idBuku);
99             stat.executeUpdate();
100        } catch (SQLException e) {
101            e.printStackTrace();
102        }
103    }
104
105    public static int getStokById(String idBuku) {
106        int stok = 0;
107        String sql = "SELECT stok FROM buku WHERE id = ?";
108        try (Connection conn = DatabaseConnection.getConnection();
109              PreparedStatement stat = conn.prepareStatement(sql)) {
110            stat.setString(1, idBuku);
111            try (ResultSet rs = stat.executeQuery()) {
112                if (rs.next()) {
113                    stok = rs.getInt("stok");
114                }
115            }
116        } catch (SQLException e) {
117            e.printStackTrace();
118        }
119        return stok;
120    }
121
122}
```

- **PeminjamanDAO.java**

```

1 package dao;
2
3 import db.DatabaseConnection;
4 import model.Peminjaman;
5
6 import java.sql.*;
7 import java.util.ArrayList;
8 import java.util.List;
9
10 public class PeminjamanDAO {
11     public static void tambahPeminjaman(Peminjaman p) {
12         String sql = "INSERT INTO peminjaman (id_anggota, id_buku, tanggal_pinjam, tanggal_kembali, status, status_pembayaran_denda) VALUES (?, ?, ?, ?, ?, ?)";
13         try (Connection conn = DatabaseConnection.getConnection();
14             PreparedStatement stmt = conn.prepareStatement(sql)) {
15             stmt.setInt(1, p.getIdAnggota());
16             stmt.setInt(2, p.getIdBuku());
17             stmt.setDate(3, Date.valueOf(p.getTanggalPinjam()));
18             if (p.getTanggalKembali() != null) {
19                 stmt.setDate(4, Date.valueOf(p.getTanggalKembali()));
20             } else {
21                 stmt.setNull(4, java.sql.Types.DATE);
22             }
23             stmt.setString(5, p.getStatus());
24             stmt.setString(6, p.getStatusPembayaranDenda());
25             stmt.executeUpdate();
26         } catch (SQLException e) {
27             e.printStackTrace();
28         }
29     }
30
31     public static List<Peminjaman> getAll() {
32         List<Peminjaman> list = new ArrayList<>();
33         String sql = "SELECT * FROM peminjaman";
34         try (Connection conn = DatabaseConnection.getConnection();
35             PreparedStatement stmt = conn.prepareStatement(sql)) {
36             ResultSet rs = stmt.executeQuery();
37             while (rs.next()) {
38                 Peminjaman p = new Peminjaman(rs.getInt("id_anggota"), rs.getInt("id_buku"),
39                     rs.getDate("tanggal_pinjam").toLocalDate(),
40                     rs.getDate("tanggal_kembali") != null ? rs.getDate("tanggal_kembali").toLocalDate() : null);
41                 p.setId(rs.getInt("id"));
42                 p.setStatus(rs.getString("status"));
43                 p.setStatusPembayaranDenda(rs.getString("status_pembayaran_denda"));
44                 list.add(p);
45             }
46         } catch (SQLException e) {
47             e.printStackTrace();
48         }
49         return list;
50     }
51
52     public static List<Peminjaman> getByIdAnggota(int idAnggota) {
53         List<Peminjaman> list = new ArrayList<>();
54         String sql = "SELECT * FROM peminjaman WHERE id_anggota = ?";
55         try (Connection conn = DatabaseConnection.getConnection();
56             PreparedStatement stmt = conn.prepareStatement(sql)) {
57             stmt.setInt(1, idAnggota);
58             try (ResultSet rs = stmt.executeQuery()) {
59                 while (rs.next()) {
60                     Peminjaman p = new Peminjaman(rs.getInt("id_anggota"), rs.getInt("id_buku"),
61                         rs.getDate("tanggal_pinjam").toLocalDate(),
62                         rs.getDate("tanggal_kembali") != null ? rs.getDate("tanggal_kembali").toLocalDate() : null);
63                     p.setId(rs.getInt("id"));
64                     p.setStatus(rs.getString("status"));
65                     list.add(p);
66                 }
67             } catch (SQLException e) {
68                 e.printStackTrace();
69             }
70         }
71         return list;
72     }
73
74     public static void hapusPeminjaman(int id) {
75         String sql = "DELETE FROM peminjaman WHERE id = ?";
76         try (Connection conn = DatabaseConnection.getConnection();
77             PreparedStatement stmt = conn.prepareStatement(sql)) {
78             stmt.setInt(1, id);
79             stmt.executeUpdate();
80         } catch (SQLException e) {
81             e.printStackTrace();
82         }
83     }
84
85     public static void updateStatusPengembalian(int id) {
86         String sql = "UPDATE peminjaman SET status = 'dikembalikan' WHERE id = ?";
87         try (Connection conn = DatabaseConnection.getConnection();
88             PreparedStatement stmt = conn.prepareStatement(sql)) {
89             stmt.setInt(1, id);
90             stmt.executeUpdate();
91         } catch (SQLException e) {
92             e.printStackTrace();
93         }
94     }
95
96     public static void updateStatusAndTanggalKembali(int id, java.time.LocalDate tanggalKembali) {
97         String sql = "UPDATE peminjaman SET status = 'dikembalikan', tanggal_kembali = ? WHERE id = ?";
98         try (Connection conn = DatabaseConnection.getConnection();
99             PreparedStatement stmt = conn.prepareStatement(sql)) {
100             stmt.setDate(1, java.sql.Date.valueOf(tanggalKembali));
101             stmt.setInt(2, id);
102             stmt.executeUpdate();
103         } catch (SQLException e) {
104             e.printStackTrace();
105         }
106     }
107
108     public static void updateStatusPembayaranDenda(int id, String statusPembayaranDenda) {
109         String sql = "UPDATE peminjaman SET status_pembayaran_denda = ? WHERE id = ?";
110         try (Connection conn = DatabaseConnection.getConnection();
111             PreparedStatement stmt = conn.prepareStatement(sql)) {
112             stmt.setString(1, statusPembayaranDenda);
113             stmt.setInt(2, id);
114             stmt.executeUpdate();
115         } catch (SQLException e) {
116             e.printStackTrace();
117         }
118     }
119 }
120 }
```

- UserDAO.java

```
1 package dao;
2
3 import model.User;
4 import db.DatabaseConnection;
5
6 import java.sql.*;
7
8 public class UserDAO {
9     public static User login(String username, String password) {
10         String sql = "SELECT * FROM user WHERE username = ? AND password = ?";
11         try (Connection conn = DatabaseConnection.getConnection();
12              PreparedStatement stmt = conn.prepareStatement(sql)) {
13             stmt.setString(1, username);
14             stmt.setString(2, password);
15             ResultSet rs = stmt.executeQuery();
16
17             if (rs.next()) {
18                 String role = rs.getString("role");
19                 return new User(username, password, role);
20             }
21         } catch (SQLException e) {
22             e.printStackTrace();
23         }
24         return null; // login gagal
25     }
26
27     public static void tambahUser (User user) {
28         String sql = "INSERT INTO user (username, password, role) VALUES (?, ?, ?)";
29         try (Connection conn = DatabaseConnection.getConnection();
30              PreparedStatement stmt = conn.prepareStatement(sql)) {
31             stmt.setString(1, user.getUsername());
32             stmt.setString(2, user.getPassword());
33             stmt.setString(3, user.getRole());
34             stmt.executeUpdate();
35         } catch (SQLException e) {
36             e.printStackTrace();
37         }
38     }
39 }
40
```

- **Anggota.java**

```

1 package ddp;
2
3 import model.Anggota;
4 import model.Buku;
5 import db.DatabaseConnection;
6
7 import java.sql.*;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 public class BukuDAO {
12     public static void simpan(Buku buku) {
13         String sql = "INSERT INTO buku (judul, penulis, penerbit, tahun_terbit, stok) VALUES (?, ?, ?, ?, ?)";
14         try (Connection conn = DatabaseConnection.getConnection();
15              PreparedStatement stmt = conn.prepareStatement(sql)) {
16             stmt.setString(1, buku.getJudul());
17             stmt.setString(2, buku.getPenulis());
18             stmt.setString(3, buku.getPenerbit());
19             stmt.setInt(4, buku.getTahunTerbit());
20             stmt.setInt(5, buku.getStok());
21             stmt.executeUpdate();
22         } catch (SQLException e) {
23             e.printStackTrace();
24         }
25     }
26
27     public static List<Buku> getAll() {
28         List<Buku> list = new ArrayList<>();
29         String sql = "SELECT * FROM buku";
30         try (Connection conn = DatabaseConnection.getConnection();
31              PreparedStatement stmt = conn.prepareStatement(sql);
32              ResultSet rs = stmt.executeQuery()) {
33             while (rs.next()) {
34                 Buku buku = new Buku(
35                     rs.getString("id"),
36                     rs.getString("judul"),
37                     rs.getString("penulis"),
38                     rs.getString("penerbit"),
39                     rs.getInt("tahun_terbit"),
40                     rs.getInt("stok")
41                 );
42                 list.add(buku);
43             }
44         } catch (SQLException e) {
45             e.printStackTrace();
46         }
47     }
48     return list;
49 }
50
51     public static void hapusBuku(String id) {
52         String sql = "DELETE FROM buku WHERE id = ?";
53         try (Connection conn = DatabaseConnection.getConnection();
54              PreparedStatement stmt = conn.prepareStatement(sql)) {
55             stmt.setString(1, id);
56             stmt.executeUpdate();
57         } catch (SQLException e) {
58             e.printStackTrace();
59         }
60     }
61
62     public static void tambahBuku(Buku o) {
63         String sql = "INSERT INTO buku (id, judul, penulis, penerbit, tahun_terbit, stok) VALUES (?, ?, ?, ?, ?, ?)";
64         try (Connection conn = DatabaseConnection.getConnection();
65              PreparedStatement stmt = conn.prepareStatement(sql)) {
66             stmt.setString(1, o.getId());
67             stmt.setString(2, o.getJudul());
68             stmt.setString(3, o.getPenulis());
69             stmt.setString(4, o.getPenerbit());
70             stmt.setInt(5, o.getTahunTerbit());
71             stmt.setInt(6, o.getStok());
72             stmt.executeUpdate();
73         } catch (SQLException e) {
74             e.printStackTrace();
75         }
76     }
77 }
78
79     public static void updateBuku(Buku o) {
80         String sql = "UPDATE buku SET judul = ?, penulis = ?, penerbit = ?, tahun_terbit = ?, stok = ? WHERE id = ?";
81         try (Connection conn = DatabaseConnection.getConnection();
82              PreparedStatement stmt = conn.prepareStatement(sql)) {
83
84             stmt.setString(1, o.getJudul());
85             stmt.setString(2, o.getPenulis());
86             stmt.setString(3, o.getPenerbit());
87             stmt.setInt(4, o.getTahunTerbit());
88             stmt.setInt(5, o.getStok());
89             stmt.setString(6, o.getId());
90             stmt.executeUpdate();
91
92         } catch (SQLException e) {
93             e.printStackTrace();
94         }
95     }
96
97     public static void updateStok(String idBuku, int stokBaru) {
98         String sql = "UPDATE buku SET stok = ? WHERE id = ?";
99         try (Connection conn = DatabaseConnection.getConnection();
100            PreparedStatement stmt = conn.prepareStatement(sql)) {
101            stmt.setInt(1, stokBaru);
102            stmt.setString(2, idBuku);
103            stmt.executeUpdate();
104        } catch (SQLException e) {
105            e.printStackTrace();
106        }
107    }
108
109    public static int getStokById(String idBuku) {
110        int stok = 0;
111        String sql = "SELECT stok FROM buku WHERE id = ?";
112        try (Connection conn = DatabaseConnection.getConnection();
113             PreparedStatement stmt = conn.prepareStatement(sql)) {
114            stmt.setString(1, idBuku);
115            try (ResultSet rs = stmt.executeQuery()) {
116                if (rs.next()) {
117                    stok = rs.getInt("stok");
118                }
119            }
120        } catch (SQLException e) {
121            e.printStackTrace();
122        }
123        return stok;
124    }
125
126
127 }

```

- **Buku.java**

```
1 // Source code is decompiled from a .class file using FernFlower decompiler.
2 package model;
3
4 public class Buku {
5     private String id;
6     private String judul;
7     private String penulis;
8     private String penerbit;
9     private int tahunTerbit;
10    private int stok;
11
12    public Buku(String var1, String var2, String var3, String var4, int var5, int var6) {
13        this.id = var1;
14        this.judul = var2;
15        this.penulis = var3;
16        this.penerbit = var4;
17        this.tahunTerbit = var5;
18        this.stok = var6;
19    }
20
21    public String getId() {
22        return this.id;
23    }
24
25    public void setId(String var1) {
26        this.id = var1;
27    }
28
29    public String getJudul() {
30        return this.judul;
31    }
32
33    public void setJudul(String var1) {
34        this.judul = var1;
35    }
36
37    public String getPenulis() {
38        return this.penulis;
39    }
40
41    public void setPenulis(String var1) {
42        this.penulis = var1;
43    }
44
45    public String getPenerbit() {
46        return this.penerbit;
47    }
48
49    public void setPenerbit(String var1) {
50        this.penerbit = var1;
51    }
52
53    public int getTahunTerbit() {
54        return this.tahunTerbit;
55    }
56
57    public void setTahunTerbit(int var1) {
58        this.tahunTerbit = var1;
59    }
60
61    public int getStok() {
62        return this.stok;
63    }
64
65    public void setStok(int var1) {
66        this.stok = var1;
67    }
68 }
69 }
```

- **Peminjaman.java**

```
1 package model;
2
3 import java.time.LocalDate;
4
5 public class Peminjaman {
6     private int id;
7     private int idAnggota;
8     private int idBuku;
9     private LocalDate tanggalPinjam;
10    private LocalDate tanggalKembali;
11    private String status;
12    private String statusPembayaranDenda;
13
14    public Peminjaman(int idAnggota, int idBuku, LocalDate tanggalPinjam, LocalDate tanggalKembali) {
15        this.idAnggota = idAnggota;
16        this.idBuku = idBuku;
17        this.tanggalPinjam = tanggalPinjam;
18        this.tanggalKembali = tanggalKembali;
19        this.status = "dipinjam"; // Status default
20        this.statusPembayaranDenda = "Belum Bayar"; // Default payment status
21    }
22
23    // Getter dan Setter
24    public int getId() { return id; }
25    public void setId(int id) { this.id = id; }
26    public int getIdAnggota() { return idAnggota; }
27    public void setIdAnggota(int idAnggota) { this.idAnggota = idAnggota; }
28    public int getIdBuku() { return idBuku; }
29    public void setIdBuku(int idBuku) { this.idBuku = idBuku; }
30    public LocalDate getTanggalPinjam() { return tanggalPinjam; }
31    public void setTanggalPinjam(LocalDate tanggalPinjam) { this.tanggalPinjam = tanggalPinjam; }
32    public LocalDate getTanggalKembali() { return tanggalKembali; }
33    public void setTanggalKembali(LocalDate tanggalKembali) { this.tanggalKembali = tanggalKembali; }
34    public String getStatus() { return status; }
35    public void setStatus(String status) { this.status = status; }
36    public String getStatusPembayaranDenda() { return statusPembayaranDenda; }
37    public void setStatusPembayaranDenda(String statusPembayaranDenda) { this.statusPembayaranDenda = statusPembayaranDenda; }
38 }
```

- User.java



The screenshot shows a code editor window with a dark theme. At the top left, there are three colored circular icons: red, yellow, and green. The code editor displays the following Java code:

```
1 package model;
2
3 public class User {
4     private String username;
5     private String password;
6     private String role;
7
8     public User(String username, String password, String role) {
9         this.username = username;
10        this.password = password;
11        this.role = role;
12    }
13
14    // Getter dan Setter
15    public String getUsername() { return username; }
16    public void setUsername(String username) { this.username = username; }
17    public String getPassword() { return password; }
18    public void setPassword(String password) { this.password = password; }
19    public String getRole() { return role; }
20    public void setRole(String role) { this.role = role; }
21 }
```

## ● AnggotaGUI.java

```

1 package view;
2
3 import model.Anggota;
4 import dao.AnggotaDAO;
5
6 import java.awt.*;
7 import javax.swing.*;
8 import javax.swing.table.DefaultTableModel;
9 import java.awt.event.*;
10 import java.util.List;
11
12 public class AnggotaGUI extends JFrame {
13     private TableTable;
14     private DefaultTableModel tableModel;
15     private JTextField tName, tJurusun, tEmail;
16     private JFrame parentDashboard; // Simpan referensi parent dashboard
17
18     public AnggotaGUI(JFrame parent) {
19         this.parentDashboard = parent; // Simpan referensi parent
20
21         setTitle("Anggota");
22         setLayout(null);
23         setLocationRelativeTo(null);
24         setFocusable(true);
25         setFocusTraversalKeys(null);
26
27         JPanel panelForm = new JPanel(new GridLayout(5, 2, 10, 10));
28         panelForm.setBoarder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
29
30         panelForm.add(new JLabel("NIM"));
31         tName = new JTextField();
32         panelForm.add(tName);
33
34         panelForm.add(new JLabel("Nama"));
35         tName = new JTextField();
36         panelForm.add(tName);
37
38         panelForm.add(new JLabel("Jurusan"));
39         tJurusun = new JTextField();
40         panelForm.add(tJurusun);
41
42         panelForm.add(new JLabel("Email"));
43         tEmail = new JTextField();
44         panelForm.add(tEmail);
45
46         JButton btnTambah = new JButton("Tambah");
47         JButton btnUpdate = new JButton("Update");
48         JButton btnDelete = new JButton("Hapus");
49         JButton btnSearch = new JButton("Cari");
50
51         JPanel panelButtons = new JPanel();
52         panelButtons.add(btnTambah);
53         panelButtons.add(btnUpdate);
54         panelButtons.add(btnDelete);
55         panelButtons.add(btnSearch);
56
57         String[] kolom = {"NIM", "Nama", "Jurusan", "Email"};
58         table = new JTable(tableModel);
59         table.setFisrtRow(true);
60         table.setRowHeight(20);
61         table.setGridColor(Color.GRAY);
62
63         JScrollPane scrollPane = new JScrollPane(table);
64
65         setLayout(new BorderLayout());
66         add(panelForm, BorderLayout.NORTH);
67         add(scrollPane, BorderLayout.CENTER);
68         add(panelButtons, BorderLayout.SOUTH);
69
70         panelButtons.addActionListener(e -> {
71             if (e.getSource() == btnTambah) {
72                 panelDashboard.setVisible(false); // Tampilkan dashboard
73                 panelDashboard.setVisible(true); // Tampilkan dashboard
74             }
75         });
76
77         table.addMouseListener(new MouseAdapter() {
78             public void mouseClicked(MouseEvent e) {
79                 int row = table.getSelectedRow();
80                 if (row > -1) {
81                     tName.setText(tableModel.getValueAt(row, 0).toString());
82                     tName.setText(tableModel.getValueAt(row, 1).toString());
83                     tJurusun.setText(tableModel.getValueAt(row, 2).toString());
84                     tEmail.setText(tableModel.getValueAt(row, 3).toString());
85                     tName.setEditable(true);
86                 }
87             }
88         });
89
90         setVisible(true);
91
92         private void loadData() {
93             tableModel.setRowCount(0);
94             listAnggota = listAnggotaDAO.getAll();
95             for (Anggota data : listAnggota) {
96                 Object[] data = {data.getNim(), data.getNama(), data.getJurusan(), data.getEmail()};
97                 tableModel.addRow(data);
98             }
99         }
100
101         private void tambahAnggota() {
102             String nim = tName.getText().trim();
103             String nama = tName.getText().trim();
104             String jurusan = tJurusun.getText().trim();
105             String email = tEmail.getText().trim();
106
107             if (nim.isEmpty() || nama.isEmpty()) {
108                 JOptionPane.showMessageDialog(this, "NIM dan Nama wajib diisi", "Error", JOptionPane.ERROR_MESSAGE);
109                 return;
110             }
111
112             Anggota anggota = new Anggota(nim, nama, jurusan, email);
113             AnggotaDAO tambahAnggota(Anggota);
114             JOptionPane.showMessageDialog(this, "Anggota berhasil ditambahkan");
115             clearForm();
116             loadData();
117         }
118
119         private void updateAnggota() {
120             String nim = tName.getText().trim();
121             String nama = tName.getText().trim();
122             String jurusan = tJurusun.getText().trim();
123             String email = tEmail.getText().trim();
124
125             if (nim.isEmpty() || nama.isEmpty()) {
126                 JOptionPane.showMessageDialog(this, "NIM dan Nama wajib diisi", "Error", JOptionPane.ERROR_MESSAGE);
127             }
128
129             Anggota a = new Anggota(nim, nama, jurusan, email);
130             AnggotaDAO updateAnggota(a);
131             JOptionPane.showMessageDialog(this, "Anggota berhasil diupdate");
132             clearForm();
133             loadData();
134             tName.setEditable(true);
135         }
136
137         private void hapusAnggota() {
138             String nim = tName.getText().trim();
139             if (nim.isEmpty()) {
140                 JOptionPane.showMessageDialog(this, "Pilih anggota yang ingin dihapus!", "Error", JOptionPane.ERROR_MESSAGE);
141             }
142             else {
143                 int confirm = JOptionPane.showConfirmDialog(this, "Apakah anda yakin ingin menghapus anggota dengan NIM " + nim + "?", "Konfirmasi", JOptionPane.YES_NO_OPTION);
144                 if (confirm == JOptionPane.YES_OPTION) {
145                     AnggotaDAO hapusAnggota(nim);
146                     JOptionPane.showMessageDialog(this, "Anggota berhasil dihapus!");
147                     clearForm();
148                     loadData();
149                     tName.setEditable(true);
150                 }
151             }
152         }
153
154         private void clearForm() {
155             tName.setText("");
156             tName.setEditable(true);
157             tJurusun.setText("");
158             tEmail.setText("");
159             tEmail.setEditable(true);
160             tName.setCaretPosition(0);
161         }
162
163         public static void main(String[] args) {
164             SwingUtilities.invokeLater(() -> new AnggotaGUI(new DashboardPustakawan("puslatawan"))); // Untuk pemungulan
165         }
166     }

```

- BukuGUI.java



- **DashboardGUI.java**

```
 1 package view;
 2
 3 import javax.swing.*;
 4
 5 import model.Peminjaman;
 6
 7 import java.awt.*;
 8 import java.awt.event.ActionEvent;
 9
10 public class DashboardGUI extends JFrame {
11     private String userRole;
12
13     private JButton btnKelolaAnggota;
14     private JButton btnKelolaBuku;
15     private JButton btnPeminjaman;
16     private JButton btnPengembalian;
17
18     public DashboardGUI(String role) {
19         this.userRole = role;
20
21         setTitle("Dashboard Perpustakaan");
22         setSize(400, 300);
23         setLocationRelativeTo(null);
24         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25         setLayout(new GridLayout(4, 1, 10, 10));
26
27         btnKelolaAnggota = new JButton("Kelola Anggota");
28         btnKelolaBuku = new JButton("Kelola Buku");
29         btnPeminjaman = new JButton("Peminjaman Buku");
30         btnPengembalian = new JButton("Pengembalian Buku");
31
32         add(btnKelolaAnggota);
33         add(btnKelolaBuku);
34         add(btnPeminjaman);
35         add(btnPengembalian);
36
37         btnKelolaAnggota.addActionListener((ActionEvent e) -> {
38             this.setVisible(false); // Sembunyikan Dashboard saat buka GUI baru
39             new AnggotaGUI(this);
40         });
41         btnKelolaBuku.addActionListener((ActionEvent e) -> {
42             this.setVisible(false);
43             new BukuGUI(this, userRole);
44         });
45         btnPeminjaman.addActionListener((ActionEvent e) -> {
46             this.setVisible(false);
47             new PeminjamanGUI(this);
48         });
49         btnPengembalian.addActionListener((ActionEvent e) -> {
50             this.setVisible(false);
51             new PengembalianGUI(this);
52         });
53
54         setVisible(true);
55     }
56
57     public static void main(String[] args) {
58         new DashboardGUI("pustakawan");
59     }
60 }
61
```

- **DashboardPengunjung.java**

```
1 package view;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class DashboardPengunjung extends JFrame {
7     private String userRole;
8     private int currentUserId; // Add current user's anggota ID
9
10    public DashboardPengunjung(String role, int currentUserId) {
11        this.userRole = role;
12        this.currentUserId = currentUserId;
13
14        setTitle("Dashboard Pengunjung");
15        // setSize(400, 250);
16        setDefaultCloseOperation(EXIT_ON_CLOSE);
17        setLocationRelativeTo(null);
18        setExtendedState(JFrame.MAXIMIZED_BOTH);
19
20        JPanel panel = new JPanel(new GridLayout(4, 1, 10, 10));
21        panel.setBorder(BorderFactory.createEmptyBorder(30, 50, 30, 50));
22
23        JLabel label = new JLabel("Selamat Datang, Pengunjung", JLabel.CENTER);
24        label.setFont(new Font("Arial", Font.BOLD, 16));
25
26        JButton btnLihatBuku = new JButton("Lihat Daftar Buku");
27        JButton btnRiwayat = new JButton("Riwayat Peminjaman");
28        JButton btnLogout = new JButton("Logout");
29
30        // Set preferred sizes for buttons
31        Dimension buttonSize = new Dimension(200, 50);
32        btnLihatBuku.setPreferredSize(buttonSize);
33        btnRiwayat.setPreferredSize(buttonSize);
34        btnLogout.setPreferredSize(buttonSize);
35
36        panel.add(label);
37        panel.add(btnLihatBuku);
38        panel.add(btnRiwayat);
39        panel.add(btnLogout);
40
41        add(panel);
42
43        btnLihatBuku.addActionListener(e -> {
44            BukuGUI bukuGUI = new BukuGUI(this, userRole); // Kirim referensi ini dan role
45            bukuGUI.setVisible(true); // Menampilkan BukuGUI
46            this.setVisible(false); // Menyembunyikan DashboardPengunjung
47        });
48
49        btnRiwayat.addActionListener(e -> {
50            RiwayatGUI riwayatGUI = new RiwayatGUI(this, currentUserId); // Kirim referensi ini dan currentUserID
51            riwayatGUI.setVisible(true); // Menampilkan RiwayatGUI
52            this.setVisible(false); // Menyembunyikan DashboardPengunjung
53        });
54
55        btnLogout.addActionListener(e -> {
56            new LoginGUI(); // Membuka LoginGUI
57            this.dispose(); // Menutup DashboardPengunjung
58        });
59
60        setVisible(true);
61    }
62 }
63 }
```

- **DashboardPustakawan.java**



```

1 package view;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class DashboardPustakawan extends JFrame {
7     private String userRole;
8
9     public DashboardPustakawan(String role) {
10         this.userRole = role;
11
12         setTitle("Dashboard Pustakawan");
13         // setSize(400, 400);
14         setLocationRelativeTo(null);
15         setDefaultCloseOperation(EXIT_ON_CLOSE);
16         setExtendedState(JFrame.MAXIMIZED_BOTH);
17
18         JPanel panel = new JPanel(new GridLayout(8, 1, 10, 10));
19         panel.setBorder(BorderFactory.createEmptyBorder(20, 40, 20, 40));
20
21         JLabel label = new JLabel("Selamat Datang, Pustakawan", JLabel.CENTER);
22         label.setFont(new Font("Arial", Font.BOLD, 16));
23
24         JButton btnBuku = new JButton("Kelola Buku");
25         JButton btnAnggota = new JButton("Kelola Anggota");
26         JButton btnPeminjaman = new JButton("Kelola Peminjaman");
27         JButton btnPengembalian = new JButton("Pengembalian Buku");
28         JButton btnDenda = new JButton("Laporan Denda");
29         JButton btnLogout = new JButton("Logout");
30
31         panel.add(label);
32         panel.add(btnBuku);
33         panel.add(btnAnggota);
34         panel.add(btnPeminjaman);
35         panel.add(btnPengembalian);
36         panel.add(btnDenda);
37         panel.add(btnLogout);
38
39         add(panel);
40
41         btnBuku.addActionListener(e -> {
42             BukuGUI bukuGUI = new BukuGUI(this, userRole); // Kirim referensi ini dan role
43             bukuGUI.setVisible(true); // Menampilkan BukuGUI
44             this.setVisible(false); // Menyembunyikan DashboardPustakawan
45         });
46
47         btnAnggota.addActionListener(e -> {
48             AnggotaGUI anggotaGUI = new AnggotaGUI(this); // Kirim referensi ini
49             anggotaGUI.setVisible(true); // Menampilkan AnggotaGUI
50             this.setVisible(false); // Menyembunyikan DashboardPustakawan
51         });
52
53         btnPeminjaman.addActionListener(e -> {
54             PeminjamanGUI peminjamanGUI = new PeminjamanGUI(this); // Kirim referensi ini
55             peminjamanGUI.setVisible(true); // Menampilkan PeminjamanGUI
56             this.setVisible(false); // Menyembunyikan DashboardPustakawan
57         });
58
59         btnPengembalian.addActionListener(e -> {
60             PengembalianGUI pengembalianGUI = new PengembalianGUI(this); // Kirim referensi ini
61             pengembalianGUI.setVisible(true); // Menampilkan PengembalianGUI
62             this.setVisible(false); // Menyembunyikan DashboardPustakawan
63         });
64
65         btnDenda.addActionListener(e -> {
66             DendaGUI dendaGUI = new DendaGUI(this); // Kirim referensi ini
67             dendaGUI.setVisible(true); // Menampilkan DendaGUI
68             this.setVisible(false); // Menyembunyikan DashboardPustakawan
69         });
70
71         btnLogout.addActionListener(e -> {
72             new LoginGUI(); // Membuka LoginGUI
73             this.dispose(); // Menutup DashboardPustakawan
74         });
75
76         setVisible(true);
77     }
78
79     public static void main(String[] args) {
80         SwingUtilities.invokeLater(() -> new DashboardPustakawan("pustakawan")); // Menjalankan aplikasi dengan role pustakawan
81     }
82 }
83

```

- DendaGUI.java

```

1 package view;
2
3 import dao.PeminjamanDAO;
4 import model.Peminjaman;
5
6 import javax.swing.*;
7 import javax.swing.table.DefaultTableModel;
8 import java.awt.*;
9 import java.time.LocalDate;
10 import java.time.temporal.ChronoUnit;
11 import java.util.List;
12 import java.util.Map;
13 import java.util.stream.Collectors;
14
15 public class DendaGUI extends JFrame {
16     private JTable table;
17     private JFrame parentDashboard; // Simpan referensi parent dashboard
18
19     public DendaGUI(JFrame parent) {
20         this.parentDashboard = parent; // Simpan referensi parent
21
22         setTitle("Laporan Denda");
23         setSize(700, 400);
24         setLocationRelativeTo(null);
25         setDefaultCloseOperation(EXIT_ON_CLOSE);
26
27         table = new JTable();
28         JScrollPane scrollPane = new JScrollPane(table);
29
30         JButton btnKembali = new JButton("Kembali");
31         btnKembali.addActionListener(e -> {
32             this.dispose(); // Tutup DendaGUI
33             parentDashboard.setVisible(true); // Tampilkan dashboard induk
34         });
35
36         loadDenda();
37
38         setLayout(new BorderLayout());
39         add(scrollPane, BorderLayout.CENTER);
40         add(btnKembali, BorderLayout.SOUTH);
41
42         setVisible(true);
43     }
44
45     private void loadDenda() {
46         List<Peminjaman> list = PeminjamanDAO.getAll();
47         List<model.Buku> bukuList = dao.BukuDAO.getAll();
48         List<model.Anggota> anggotaList = new dao.AnggotaDAO().getAll();
49
50         Map<Integer, model.Buku> bukuMap = bukuList.stream()
51             .collect(Collectors.toMap(b -> Integer.parseInt(b.getId()), b -> b));
52         Map<Integer, model.Anggota> anggotaMap = anggotaList.stream()
53             .collect(Collectors.toMap(model.Anggota::getId, a -> a));
54
55         String[] columnNames = {"ID Peminjaman", "ID Anggota", "Judul Buku", "Jumlah Denda", "Status Pembayaran"};
56         DefaultTableModel model = new DefaultTableModel(columnNames, 0) {
57             @Override
58             public boolean isCellEditable(int row, int column) {
59                 // Only "Status Pembayaran" column is editable
60                 return column == 4;
61             }
62         };
63
64         for (Peminjaman p : list) {
65             if (p.getTanggalKembali() != null) {
66                 long daysLate = ChronoUnit.DAYS.between(p.getTanggalKembali(), LocalDate.now());
67                 if (daysLate > 0) {
68                     int fine = (int) daysLate * 5000;
69                     String judulBuku = bukuMap.containsKey(p.getIdBuku()) ? bukuMap.get(p.getIdBuku()).getJudul() : "Unknown";
70                     String statusPembayaran = p.getStatusPembayaranDenda() != null ? p.getStatusPembayaranDenda() : "Belum Bayar";
71                     model.addRow(new Object[]{
72                         p.getId(),
73                         p.getIdAnggota(),
74                         judulBuku,
75                         fine,
76                         statusPembayaran
77                     });
78                 }
79             }
80         }
81         table.setModel(model);
82
83         // Add listener to update payment status on cell edit
84         table.getModel().addTableModelListener(e -> {
85             if (e.getColumn() == 4) // Status Pembayaran column
86                 int row = e.getFirstRow();
87                 int idPeminjaman = (int) table.getValueAt(row, 0);
88                 String newPassword = (String) table.getValueAt(row, 4);
89                 // Update status in database
90                 dao.PeminjamanDAO.updateStatusPembayaranDenda(idPeminjaman, newPassword);
91             });
92         }
93     }
94
95     public static void main(String[] args) {
96         SwingUtilities.invokeLater(() -> new DendaGUI(new DashboardPustakawan("pustakawan"))); // Contoh pemanggilan
97     }
98 }
99

```

## ● LoginGUI.java

```

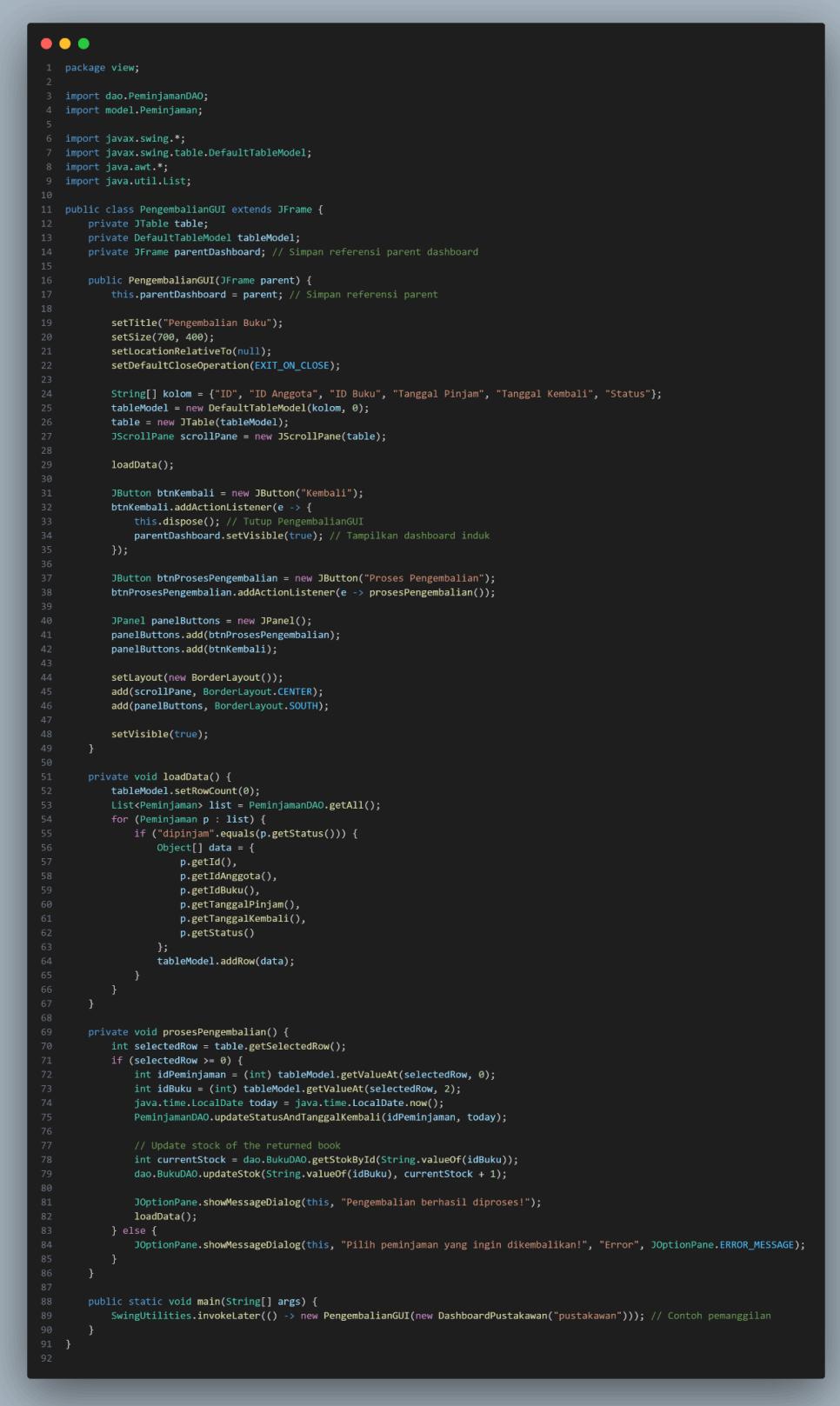
1 package view;
2
3 import dao.UserDAO;
4 import model.User;
5 import model.anggota;
6 import dao.anggotaDAO;
7
8 import javax.swing.*;
9 import java.awt.*;
10
11 public class LoginGUI extends JFrame {
12     private JTextField tfUsername;
13     private JPasswordField pfPassword;
14     private JComboBox<String> cbRole;
15
16     public LoginGUI() {
17         setTitle("Login");
18         // setSize(400, 200);
19         setExtendedState(JFrame.EXIT_ON_CLOSE);
20         setExtendedState(JFrame.MAXIMIZED_BOTH);
21         setLocationRelativeTo(null);
22         setResizable(true);
23
24         // Use GridBagLayout for flexible layout
25         JPanel panel = new JPanel(new GridBagLayout());
26         panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
27         GridBagConstraints gbc = new GridBagConstraints();
28         gbc.insets = new Insets(5, 5, 5, 5);
29         gbc.fill = GridBagConstraints.HORIZONTAL;
30
31         // Title label
32         JLabel lblTitle = new JLabel("Login");
33         lblTitle.setFont(new Font("Arial", Font.BOLD, 24));
34         lblTitle.setHorizontalAlignment(SwingConstants.CENTER);
35         gbc.gridx = 0;
36         gbc.gridy = 0;
37         gbc.gridwidth = 2;
38         panel.add(lblTitle, gbc);
39
40         // Username label
41         JLabel lblUsername = new JLabel("Username:");
42         lblUsername.setFont(new Font("Arial", Font.BOLD, 14));
43         lblUsername.setHorizontalAlignment(SwingConstants.RIGHT);
44         gbc.gridx = 0;
45         gbc.gridy = 1;
46         gbc.gridwidth = 1;
47         panel.add(lblUsername, gbc);
48
49         // Username text field
50         JTextField tfUsername = new JTextField();
51         tfUsername.setPreferredWidth(new Dimension(200, 25));
52         tfUsername.setToolTipText("Enter your username");
53         gbc.gridx = 1;
54         gbc.gridy = 1;
55         panel.add(tfUsername, gbc);
56
57         // Password label
58         JLabel lblPassword = new JLabel("Password:");
59         lblPassword.setFont(new Font("Arial", Font.BOLD, 14));
60         lblPassword.setHorizontalAlignment(SwingConstants.RIGHT);
61         gbc.gridx = 0;
62         gbc.gridy = 2;
63         panel.add(lblPassword, gbc);
64
65         // Password field
66         JPasswordField pfPassword = new JPasswordField();
67         pfPassword.setPreferredWidth(new Dimension(200, 25));
68         pfPassword.setToolTipText("Enter your password");
69         gbc.gridx = 1;
70         gbc.gridy = 2;
71         panel.add(pfPassword, gbc);
72
73         // Role label
74         JLabel lblRole = new JLabel("Login as:");
75         lblRole.setFont(new Font("Arial", Font.BOLD, 14));
76         lblRole.setHorizontalAlignment(SwingConstants.RIGHT);
77         gbc.gridx = 0;
78         gbc.gridy = 3;
79         panel.add(lblRole, gbc);
80
81         // Role combobox
82         JComboBox<String> cbRole = new JComboBox<String>[]{"Pengunjung", "Pustakawan"};
83         cbRole.setEditable(true);
84         cbRole.setPlaceholder("Select your role");
85         cbRole.setPreferredWidth(new Dimension(200, 25));
86         gbc.gridx = 1;
87         gbc.gridy = 3;
88         panel.add(cbRole, gbc);
89
90         // Login button
91         JButton btnLogin = new JButton("Login");
92         btnLogin.setFont(new Font("Arial", Font.BOLD, 14));
93         btnLogin.setForeground(Color.DARK_GRAY);
94         btnLogin.setBackground(Color.MCGRAW_HILL);
95         btnLogin.setFocusable(false);
96         btnLogin.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
97         btnLogin.setOpaque(true);
98         btnLogin.setRolloverEnabled(false);
99         btnLogin.setRolloverText("");
100        btnLogin.setRolloverTextColor(0);
101        btnLogin.setRolloverTextFont(btnLogin.getFont());
102        btnLogin.setRolloverTextSize(btnLogin.getFont().getSize());
103        btnLogin.setRolloverTextUnderline(true);
104        btnLogin.setRolloverTextVertical(true);
105        btnLogin.addActionListener(e -> prosesLogin());
106
107        setVisible(true);
108    }
109
110    private void prosesLogin() {
111        String username = tfUsername.getText().trim();
112        String password = new String(pfPassword.getPassword()).trim();
113        String selectedRole = (String) cbRole.getSelectedItem();
114
115        if (!username.equals(ignoreCase(selectedRole))) {
116            User user = UserDAO.login(username, password);
117            if (user != null) {
118                if (selectedRole.equalsIgnoreCase(user.getRole())) {
119                    JOptionPane.showMessageDialog(this, "Login berhasil sebagai " + user.getRole());
120                    dispose();
121                    new DashboardPustakawan("pustakawan");
122                } else {
123                    JOptionPane.showMessageDialog(this, "Role yang dipilih tidak sesuai dengan akun!", "Error", JOptionPane.ERROR_MESSAGE);
124                }
125            } else {
126                JOptionPane.showMessageDialog(this, "Username atau Password salah!", "Error", JOptionPane.ERROR_MESSAGE);
127            }
128        } else if ("pengunjung".equalsIgnoreCase(selectedRole)) {
129            Anggota anggota = AnggotaDAO.login(username, password);
130            if (anggota != null) {
131                JOptionPane.showMessageDialog(this, "Login berhasil sebagai Pengunjung");
132                dispose();
133                new DashboardPengunjung("pengunjung", anggota.getId());
134            } else {
135                JOptionPane.showMessageDialog(this, "Username atau Password salah!", "Error", JOptionPane.ERROR_MESSAGE);
136            }
137        } else {
138            JOptionPane.showMessageDialog(this, "Role tidak dikenali", "Error", JOptionPane.ERROR_MESSAGE);
139            new LoginGUI(); // fallback
140        }
141    }
142
143    public static void main(String[] args) {
144        SwingUtilities.invokeLater(LoginGUI::new);
145    }
146}
147
148

```

- **PeminjamanGUI.java**

```
 1 package view;
 2
 3 import dao.PeminjamanDAO;
 4 import model.Peminjaman;
 5
 6 import javax.swing.*;
 7 import javax.swing.table.DefaultTableModel;
 8 import java.awt.*;
 9 import java.util.List;
10
11 public class PeminjamanGUI extends JFrame {
12     private JTable table;
13     private DefaultTableModel tableModel;
14     private JFrame parentDashboard; // Simpan referensi parent dashboard
15
16     public PeminjamanGUI(JFrame parent) {
17         this.parentDashboard = parent; // Simpan referensi parent
18
19         setTitle("Kelola Peminjaman");
20         setSize(700, 500);
21         setLocationRelativeTo(null);
22         setDefaultCloseOperation(EXIT_ON_CLOSE);
23
24         JPanel panelTengah = new JPanel(new BorderLayout());
25         String[] kolom = {"ID Peminjaman", "ID Anggota", "ID Buku", "Tanggal Pinjam", "Tanggal Kembali", "Status"};
26         tableModel = new DefaultTableModel(kolom, 0);
27         table = new JTable(tableModel);
28         JScrollPane scrollPane = new JScrollPane(table);
29         panelTengah.add(scrollPane, BorderLayout.CENTER);
30
31         JButton btnKembali = new JButton("Kembali");
32
33         JPanel panelBawah = new JPanel();
34         panelBawah.add(btnKembali);
35
36         setLayout(new BorderLayout(10, 10));
37         add(panelTengah, BorderLayout.CENTER);
38         add(panelBawah, BorderLayout.SOUTH);
39
40         loadPeminjamanData();
41
42         btnKembali.addActionListener(e -> {
43             this.dispose(); // Tutup PeminjamanGUI
44             parentDashboard.setVisible(true); // Tampilkan dashboard induk
45         });
46
47         setVisible(true);
48     }
49
50     private void loadPeminjamanData() {
51         tableModel.setRowCount(0);
52         List<Peminjaman> list = PeminjamanDAO.getAll();
53         if (list == null || list.isEmpty()) {
54             System.out.println("No loan data found.");
55             return;
56         }
57         for (Peminjaman p : list) {
58             Object[] data = {
59                 p.getId(),
60                 p.getIdAnggota(),
61                 p.getIdBuku(),
62                 p.getTanggalPinjam() != null ? p.getTanggalPinjam() : "N/A",
63                 p.getTanggalKembali() != null ? p.getTanggalKembali() : "N/A",
64                 p.getStatus() != null ? p.getStatus() : "N/A"
65             };
66             tableModel.addRow(data);
67         }
68     }
69
70     public static void main(String[] args) {
71         SwingUtilities.invokeLater(() -> new PeminjamanGUI(new DashboardPustakawan("pustakawan"))); // Contoh pemanggilan
72     }
73 }
74 }
```

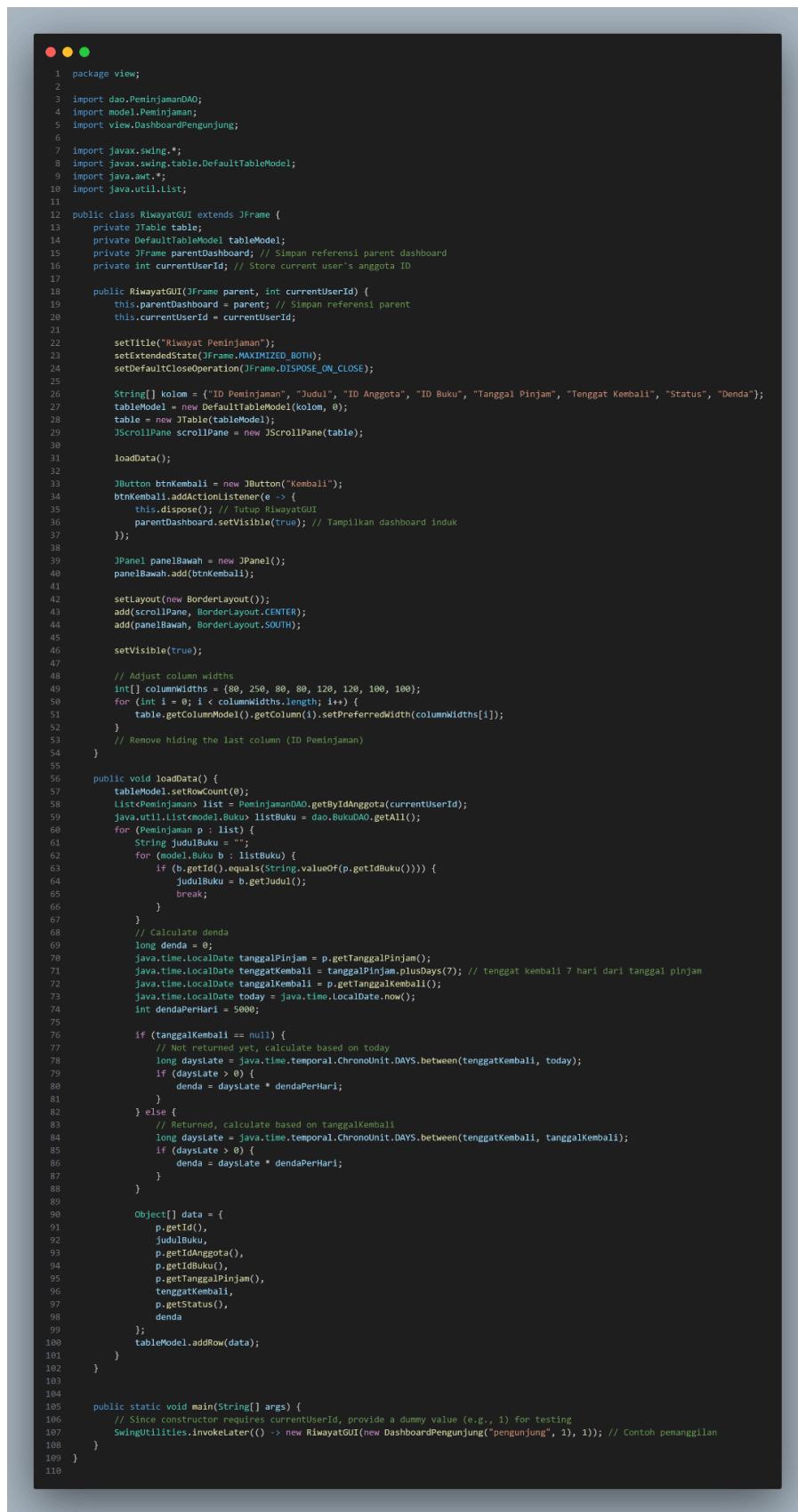
- PengembalianGUI.java



```

1 package view;
2
3 import dao.PeminjamanDAO;
4 import model.Peminjaman;
5
6 import javax.swing.*;
7 import javax.swing.table.DefaultTableModel;
8 import java.awt.*;
9 import java.util.List;
10
11 public class PengembalianGUI extends JFrame {
12     private JTable table;
13     private DefaultTableModel tableModel;
14     private JFrame parentDashboard; // Simpan referensi parent dashboard
15
16     public PengembalianGUI(JFrame parent) {
17         this.parentDashboard = parent; // Simpan referensi parent
18
19         setTitle("Pengembalian Buku");
20         setSize(700, 400);
21         setLocationRelativeTo(null);
22         setDefaultCloseOperation(EXIT_ON_CLOSE);
23
24         String[] kolom = {"ID", "ID Anggota", "ID Buku", "Tanggal Pinjam", "Tanggal Kembali", "Status"};
25         tableModel = new DefaultTableModel(kolom, 0);
26         table = new JTable(tableModel);
27         JScrollPane scrollPane = new JScrollPane(table);
28
29         loadData();
30
31         JButton btnKembali = new JButton("Kembali");
32         btnKembali.addActionListener(e -> {
33             this.dispose(); // Tutup PengembalianGUI
34             parentDashboard.setVisible(true); // Tampilkan dashboard induk
35         });
36
37         JButton btnProsesPengembalian = new JButton("Proses Pengembalian");
38         btnProsesPengembalian.addActionListener(e -> prosesPengembalian());
39
40         JPanel panelButtons = new JPanel();
41         panelButtons.add(btnProsesPengembalian);
42         panelButtons.add(btnKembali);
43
44         setLayout(new BorderLayout());
45         add(scrollPane, BorderLayout.CENTER);
46         add(panelButtons, BorderLayout.SOUTH);
47
48         setVisible(true);
49     }
50
51     private void loadData() {
52         tableModel.setRowCount(0);
53         List<Peminjaman> list = PeminjamanDAO.getAll();
54         for (Peminjaman p : list) {
55             if ("dipinjam".equals(p.getStatus())) {
56                 Object[] data = {
57                     p.getId(),
58                     p.getIdAnggota(),
59                     p.getIdBuku(),
60                     p.getTanggalPinjam(),
61                     p.getTanggalKembali(),
62                     p.getStatus()
63                 };
64                 tableModel.addRow(data);
65             }
66         }
67     }
68
69     private void prosesPengembalian() {
70         int selectedRow = table.getSelectedRow();
71         if (selectedRow >= 0) {
72             int idPeminjaman = (int) tableModel.getValueAt(selectedRow, 0);
73             int idBuku = (int) tableModel.getValueAt(selectedRow, 2);
74             java.time.LocalDate today = java.time.LocalDate.now();
75             PeminjamanDAO.updateStatusAndTanggalKembali(idPeminjaman, today);
76
77             // Update stock of the returned book
78             int currentStock = dao.BukuDAO.getStockById(String.valueOf(idBuku));
79             dao.BukuDAO.updateStock(String.valueOf(idBuku), currentStock + 1);
80
81             JOptionPane.showMessageDialog(this, "Pengembalian berhasil diproses!");
82             loadData();
83         } else {
84             JOptionPane.showMessageDialog(this, "Pilih peminjaman yang ingin dikembalikan!", "Error", JOptionPane.ERROR_MESSAGE);
85         }
86     }
87
88     public static void main(String[] args) {
89         SwingUtilities.invokeLater(() -> new PengembalianGUI(new DashboardPustakawan("pustakawan"))); // Contoh pemanggilan
90     }
91 }
92 
```

- RiwayatGUI.java



```

1 package view;
2
3 import dao.PeminjamanDAO;
4 import model.Peminjaman;
5 import view.DashboardPengunjung;
6
7 import javax.swing.*;
8 import javax.swing.table.DefaultTableModel;
9 import java.awt.*;
10 import java.util.List;
11
12 public class RiwayatGUI extends JFrame {
13     private JTable table;
14     private DefaultTableModel tableModel;
15     private JFrame parentDashboard; // Simpan referensi parent dashboard
16     private int currentUserId; // Store current user's anggota ID
17
18     public RiwayatGUI(JFrame parent, int currentUserId) {
19         this.parentDashboard = parent; // Simpan referensi parent
20         this.currentUserId = currentUserId;
21
22         setTitle("Riwayat Peminjaman");
23         setExtendedState(JFrame.MAXIMIZED_BOTH);
24         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
25
26         String[] kolom = {"ID Peminjaman", "Judul", "ID Anggota", "ID Buku", "Tanggal Pinjam", "Tenggat Kembali", "Status", "Denda"};
27         tableModel = new DefaultTableModel(kolom, 0);
28         table = new JTable(tableModel);
29         JScrollPane scrollPane = new JScrollPane(table);
30
31         loadData();
32
33         JButton btnKembali = new JButton("Kembali");
34         btnKembali.addActionListener(e -> {
35             this.dispose(); // Tutup RiwayatGUI
36             parentDashboard.setVisible(true); // Tampilkan dashboard induk
37         });
38
39         JPanel panelBawah = new JPanel();
40         panelBawah.add(btnKembali);
41
42         setLayout(new BorderLayout());
43         add(scrollPane, BorderLayout.CENTER);
44         add(panelBawah, BorderLayout.SOUTH);
45
46         setVisible(true);
47
48         // Adjust column widths
49         int[] columnWidths = {80, 250, 80, 80, 120, 120, 100, 100};
50         for (int i = 0; i < columnWidths.length; i++) {
51             table.getColumnModel().getColumn(i).setPreferredWidth(columnWidths[i]);
52         }
53         // Remove hiding the last column (ID Peminjaman)
54     }
55
56     public void loadData() {
57         tableModel.setRowCount(0);
58         List<Peminjaman> list = PeminjamanDAO.getByIdAnggota(currentUserId);
59         java.util.List<model.Buku> listBuku = dao.BukuDAO.getAll();
60         for (Peminjaman p : list) {
61             String judulBuku = "";
62             for (model.Buku b : listBuku) {
63                 if (b.getId().equals(String.valueOf(p.getIdBuku()))) {
64                     judulBuku = b.getJudul();
65                     break;
66                 }
67             }
68             // Calculate denda
69             long denda = 0;
70             java.time.LocalDate tanggalPinjam = p.getTanggalPinjam();
71             java.time.LocalDate tenggatKembali = tanggalPinjam.plusDays(7); // tenggat kembali 7 hari dari tanggal pinjam
72             java.time.LocalDate tanggalKembali = p.getTanggalKembali();
73             java.time.LocalDate today = java.time.LocalDate.now();
74             int dendaPerHari = 5000;
75
76             if (tanggalKembali == null) {
77                 // Not returned yet, calculate based on today
78                 long daysLate = java.time.temporal.ChronoUnit.DAYS.between(tenggatKembali, today);
79                 if (daysLate > 0) {
80                     denda = daysLate * dendaPerHari;
81                 }
82             } else {
83                 // Returned, calculate based on tanggalKembali
84                 long daysLate = java.time.temporal.ChronoUnit.DAYS.between(tenggatKembali, tanggalKembali);
85                 if (daysLate > 0) {
86                     denda = daysLate * dendaPerHari;
87                 }
88             }
89
90             Object[] data = {
91                 p.getId(),
92                 judulBuku,
93                 p.getIdAnggota(),
94                 p.getIdBuku(),
95                 p.getTanggalPinjam(),
96                 tenggatKembali,
97                 p.getStatus(),
98                 denda
99             };
100            tableModel.addRow(data);
101        }
102    }
103
104    public static void main(String[] args) {
105        // Since constructor requires currentUserId, provide a dummy value (e.g., 1) for testing
106        SwingUtilities.invokeLater(() -> new RiwayatGUI(new DashboardPengunjung("pengunjung", 1), 1)); // Contoh pemanggilan
107    }
108 }
109 }
```

## 2.5 Pengujian

The screenshot shows a Windows application window with two main sections. The top section is a "Login" form with fields for Username, Password, and Login as (set to Pengunjung), and a blue "Login" button. The bottom section is a "Riwayat Peminjaman" (Borrowing History) table with columns: ID Peminjaman, Judul, ID Anggota, ID Buku, Tanggal Pinjam, Tanggal Kembali, Status, and Denda. It contains two rows of data.

ID Peminjaman	Judul	ID Anggota	ID Buku	Tanggal Pinjam	Tanggal Kembali	Status	Denda
1	Jaringan Komputer Dasar	1	1	2025-06-01	2025-06-08	dikembalikan	5000
6	Jaringan Komputer Dasar	1	1	2025-06-07	2025-06-14	dikembalikan	0

Kelola Buku

ID	Judul	Penulis	Penerbit	Tahun Terbit	Stok
1	Jaringan Komputer Dasar	Agus Setiawan	Informatika	2021	3
2	Pemrograman Python untuk Data Science	Lisa Andini	Gramedia	2022	5
3	Kecerdasan Buatan dan Aplikasinya	Taufik Hidayat	Elex Media	2023	4
4	Desain UI/UX Modern	Nina Kartika	Deepublish	2021	2
5	Laskar Pelangi	Andrea Hirata	Bentang Pustaka	2005	5
6	Bumi	Tere Liye	Gramedia Pustaka Utama	2014	5
7	Pulang	Leila S. Chudori	Kepustakaan Populer Gramedia	2012	3
8	Negeri 5 Menara	Ahmad Fuadl	Gramedia	2009	4
9	Habille & Almun	B.J. Habillé	THC Mandiri	2010	2
10	Sapiens: A Brief History of Humankind	Yuval Noah Harari	Harper	2014	2
11	Atomic Habits	James Clear	Avery	2018	2
12	Tafsir Al-Misbah	M. Quraisch Shihab	Lentera Hati	2017	3
13	Fiqih Sunnah	Sayyid Sabiq	Al-Maarrif	2002	4
14	Si坑 Nabawiyah	Ibnu Hisyam	Pustaka Al-Kautsar	2015	2
15	Sejarah Indonesia Modern 1200–2008	M.C. Ricklefs	Serambi	2008	2
16	Jejak Islam di Nusantara	Ahmad Mansur Suryanegara	Rosda	2016	1
17	Revolusi Indonesia	Soekarno	Balai Pustaka	1951	0

Pinjam Buku    Kembal

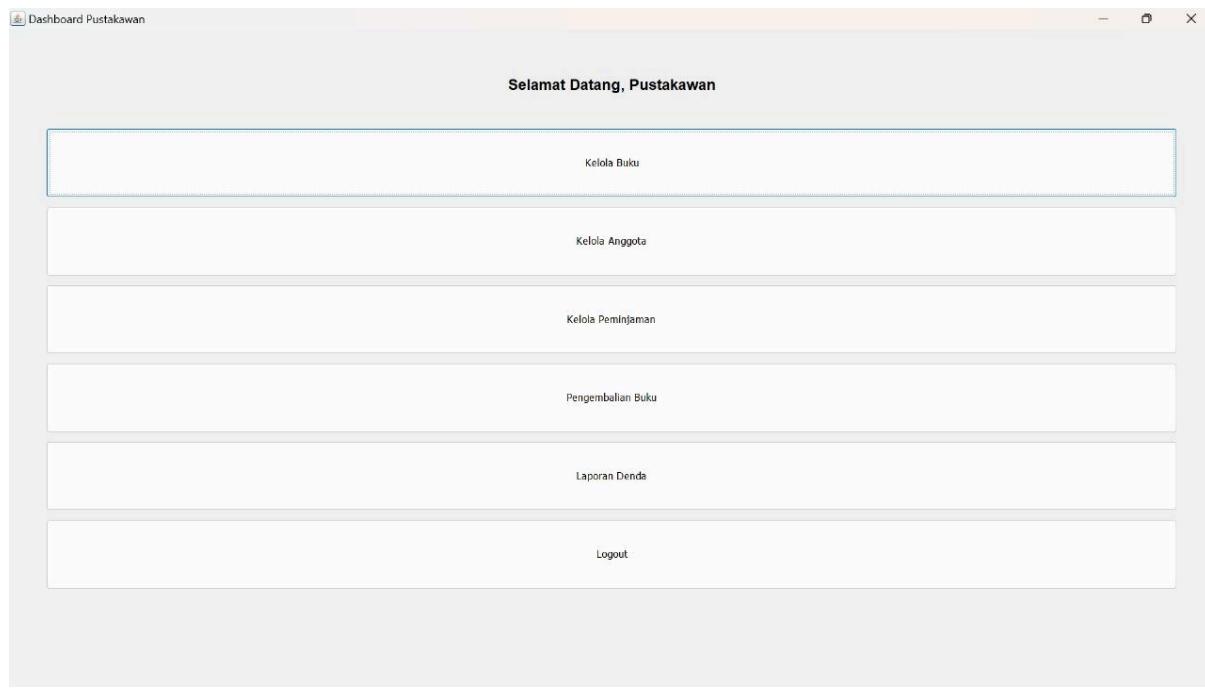
Dashboard Pengunjung

**Selamat Datang, Pengunjung**

Lihat Daftar Buku

Riwayat Peminjaman

Logout



This screenshot shows the "Kelola Buku" (Manage Books) page. At the top, there are four input fields: "ID:", "Judul:", "Penulis:", and "Penerbit:". Below these are two more fields: "Tahun Terbit:" and "Stok:". A large table lists 17 books with columns for ID, Judul, Penulis, Penerbit, Tahun Terbit, and Stok. At the bottom of the table are buttons for "Tambah", "Update", "Hapus", and "Kembali".

ID	Judul	Penulis	Penerbit	Tahun Terbit	Stok
1	Jaringan Komputer Dasar	Agus Setiawan	Informatika	2021	3
2	Pemrograman Python untuk Data Science	Lisa Andini	Gramedia	2022	5
3	Kecerdasan Buatan dan Aplikasinya	Taufik Hidayat	Elex Media	2023	4
4	Desain UI/UX Modern	Nina Kartika	Deepublish	2021	2
5	Laskar Pelangi	Andrea Hirata	Bentang Pustaka	2005	5
6	Bumi	Tere Liye	Gramedia Pustaka Utama	2014	5
7	Pulang	Lella S. Chudori	Kepustakaan Populer Gramedia	2012	3
8	Negeri 5 Menara	Ahmad Fuadli	Gramedia	2009	4
9	Habibie & Ainun	R.J. Habibie	THC Mandiri	2010	2
10	Sapiens: A Brief History of Humankind	Yuval Noah Harari	Harper	2014	2
11	Atomic Habits	James Clear	Avery	2018	2
12	Tafsir Al-Misbah	M. Qurraish Shihab	Lontara Hati	2017	3
13	Fiqih Sunnah	Sayyid Sabiq	Al-Haarif	2002	4
14	Sirah Nabawiyyah	Ibnu Hisyam	Pustaka Al-Kautsar	2015	2
15	Sejarah Indonesia Modern 1200–2008	M.C. Ricklefs	Serambi	2008	2
16	Jejak Islam di Nusantara	Ahmad Mansur Suryanegara	Rosda	2016	1
17	Revolusi Indonesia	Soekarno	Balai Pustaka	1951	0

**Kelola Anggota**

NIM:	<input type="text"/>
Nama:	<input type="text"/>
Jurusan:	<input type="text"/>
E-mail:	<input type="text"/>
Alamat:	<input type="text"/>
Kontak:	<input type="text"/>
Keterangan:	<input type="text"/>

ID Anggota	Nama	Jurusan	Email
A123456	Rizki Sembiring	Sistem Informasi	rizi@gmail.com
A123457	Ani Lestari	Sistem Informasi	ani@gmail.com
A123458	Dina Nuraini	Teknik Komputer	dina@gmail.com
A123459	Rian Saputra	Teknik Elektro	rian@gmail.com
A123460	Bati Murnhaliza	Manajemen Informatika	bati@gmail.com

[Tambah](#) [Update](#) [Hapus](#) [Kembali](#)

**Kelola Peminjaman**

ID Peminjaman	ID Anggota	ID Buku	Tanggal Pinjam	Tanggal Kembali	Status
1	1	1	2025-06-01	N/A	dikembalikan
2	2	2	2025-06-02	2025-06-05	dikembalikan
3	3	3	2025-06-03	N/A	dikembalikan
4	4	4	2025-05-30	2025-06-06	dikembalikan
5	5	5	2025-06-05	N/A	dikembalikan
6	1	1	2025-06-07	2025-06-14	dikembalikan
7	2	5	2025-06-09	N/A	dikembalikan
8	2	5	2025-06-09	2025-06-09	dikembalikan
9	2	13	2025-06-09	2025-06-09	dikembalikan

[Kembali](#)

ID Peminjaman	ID Anggota	Judul Buku	Jumlah Denda	Status Pembayaran
2	2	Programiran Python untuk Data Science	20000	Belum Bayar
4	4	Desain UI/UX Modern	15000	Belum Bayar

Kembali

## **BAB III**

## **PENUTUP**

### **3.1 Kesimpulan**

Melalui pengembangan proyek “Sistem Perpustakaan Kampus”, dapat disimpulkan bahwa sistem ini mampu menjadi solusi atas berbagai permasalahan dalam pengelolaan perpustakaan yang sebelumnya dilakukan secara manual. Penerapan konsep *Object Oriented Programming* (OOP) seperti enkapsulasi, pewarisan, dan polimorfisme terbukti efektif dalam membangun struktur program yang modular, terorganisir, dan mudah dikembangkan. Sistem ini juga berhasil membedakan peran pengguna antara pengunjung dan pustakawan dengan hak akses dan fitur yang sesuai, serta menyediakan antarmuka pengguna (GUI) yang intuitif untuk mendukung efisiensi dan kemudahan penggunaan. Selain memberikan manfaat praktis dalam pengelolaan perpustakaan, proyek ini juga memberikan pengalaman berharga bagi mahasiswa dalam memahami dan menerapkan prinsip-prinsip OOP secara nyata. Diharapkan sistem ini dapat dikembangkan lebih lanjut di masa depan agar mampu menangani kebutuhan perpustakaan kampus yang lebih kompleks secara profesional.