

LAPORAN UAS KELOMPOK 3

Disusun Guna Memenuhi

Ujian Akhir Semester

Pemrograman Berbasis Objek

Dosen Pengampu:

Taufik Ridwan, S.T., M.T



Oleh:

Asep Imam Wahyudi	(2310631250006)
M Aziz Sutawijaya	(2310631250095)
M Nugrah Adinda	(2310631250098)
Aditya Pratama Putra	(2310631250002)

KELAS 4B

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SINGAPERBANGSA KARAWANG

TAHUN 2025

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa, karena atas rahmat dan karunia Nya, kami dapat menyelesaikan laporan sistem terkait “Program Perpustakaan” ini dengan baik dan tepat waktu.

Laporan ini disusun sebagai salah satu bentuk pertanggungjawaban dan dokumentasi dari hasil pembuatan aplikasi yang menerapkan konsep *Object-Oriented Programming* (OOP) serta perancangan sistem menggunakan metode *Unified Modeling Language* (UML). Dalam proses penyusunan laporan ini, kami telah melalui tahapan-tahapan mulai dari analisis kebutuhan, perancangan sistem, implementasi kode program, hingga pengujian fitur-fitur yang telah dibuat.

Kami menyadari bahwa tersusunnya laporan ini tidak lepas dari dukungan, bantuan, dan bimbingan dari berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada, Bapak Taufik Ridwan S.T. M.T, selaku dosen pembimbing yang telah memberikan arahan dan masukan.

Kami berharap laporan ini dapat memberikan manfaat bagi pembaca, khususnya dalam memahami penerapan OOP dan UML dalam pengembangan perangkat lunak. Kritik dan saran yang membangun sangat penulis harapkan demi perbaikan di masa mendatang.

Akhir kata, semoga laporan ini dapat memberikan kontribusi positif bagi pengembangan ilmu pengetahuan dan teknologi.

Karawang, 9 Juni 2024

Kelompok 3

DAFTAR ISI

KATA PENGANTAR.....	2
DAFTAR ISI.....	3
BAB I.....	4
1.1 Latar Belakang.....	4
1.2 Rumusan Masalah.....	5
1.3 Tujuan.....	5
BAB II.....	7
2.1 Tentang Sistem.....	7
2.2 Konsep Pemrograman Berorientasi Objek (OOP) dalam Sistem Perpustakaan.....	10
2.3 Use Case Diagram.....	15
2.4 Class Diagram.....	17
2.5 Implementasi Kode PROGRAM.....	19
2.5.1 Admin.....	19
2.5.2 Anggota.....	237
2.5.3 Welcome.....	324
2.6 Graphical User Interface (GUI).....	349
2.6.1 Tampilan Admin.....	349
2.6.2 Tampilan Anggota.....	361
2.6.3 Tampilan Welcome.....	365
2.7 Pengujian.....	367
2.7.1 Pengujian Modul Admin.....	367
2.7.2 Pengujian Modul Anggota.....	368
2.7.3 Pengujian Modul Pengunjung.....	369
2.8 Hasil Pengujian.....	369
BAB III.....	370

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perpustakaan memiliki peran penting sebagai pusat literasi, sumber informasi, dan ruang belajar yang inklusif bagi masyarakat. Dalam praktiknya, banyak aktivitas perpustakaan yang melibatkan pencatatan data pengunjung, pengelolaan koleksi buku, administrasi anggota, hingga penyampaian informasi melalui pengumuman. Namun, ketika semua aktivitas ini dilakukan secara manual, efisiensi waktu dan akurasi data seringkali menjadi tantangan yang menghambat kualitas layanan.

Melihat kebutuhan tersebut, digitalisasi sistem perpustakaan menjadi langkah yang relevan dan mendesak. Teknologi dapat dihadirkan bukan hanya sebagai alat bantu, tetapi sebagai solusi untuk menciptakan sistem yang lebih tertata, cepat, dan mudah diakses. Oleh karena itu, dirancanglah sebuah aplikasi sistem informasi bernama Perpustakaan Mari Maca.

Aplikasi ini dikembangkan dengan mendukung tiga jenis aktor, yaitu Petugas, Anggota, dan Pengunjung, masing-masing dengan hak akses dan fungsionalitas yang berbeda sesuai peran mereka dalam ekosistem perpustakaan.

- Pengunjung dapat mengisi data kunjungan secara mandiri melalui sistem. Setelah itu, mereka dapat mengakses informasi seputar pengumuman terbaru dan melakukan pencarian koleksi buku. Bila tertarik menjadi anggota, pengunjung dapat datang ke petugas untuk melakukan pendaftaran.
- Petugas bertanggung jawab dalam mengelola seluruh data di dalam sistem, mulai dari data anggota, data kunjungan, koleksi buku, hingga pengumuman yang akan ditampilkan kepada publik. Petugas juga membuat akun bagi anggota baru yang telah mendaftar.
- Anggota yang telah memiliki akun dapat masuk ke sistem dan mengakses fitur yang lebih lengkap, seperti meminjam buku dan melihat status peminjaman.

Dengan hadirnya sistem informasi Perpustakaan Mari Maca, diharapkan pengelolaan perpustakaan menjadi lebih sistematis, transparan, dan efisien. Selain membantu petugas dalam

aspek administratif, sistem ini juga memberikan kenyamanan dan kemudahan akses informasi bagi pengunjung dan anggota. Penerapan teknologi ini sekaligus menjadi bagian dari upaya modernisasi layanan perpustakaan yang tetap menjaga nilai-nilai literasi dan keterbukaan akses.

1.2 Rumusan Masalah

1. Bagaimana merancang sistem informasi perpustakaan yang mampu memfasilitasi aktivitas kunjungan, pencarian buku, dan penyampaian pengumuman secara digital?
2. Bagaimana membangun sistem dengan pendekatan *Object-Oriented Programming* (OOP) yang dapat mengakomodasi peran tiga aktor utama, yaitu Petugas, Anggota, dan Pengunjung?
3. Bagaimana menerapkan perancangan sistem menggunakan *Unified Modeling Language* (UML) agar struktur aplikasi dapat dipahami dengan jelas dan terorganisir?
4. Bagaimana menguji fungsionalitas sistem agar dapat berjalan dengan baik sesuai kebutuhan masing-masing pengguna?

1.3 Tujuan

1. Untuk membangun sebuah sistem informasi perpustakaan yang terintegrasi dan efisien dalam mengelola aktivitas kunjungan, data buku, serta penyampaian informasi kepada pengguna.
2. Untuk menerapkan konsep *Object-Oriented Programming* dalam pengembangan aplikasi agar kode program lebih modular, fleksibel, dan mudah dikelola.
3. Untuk merancang sistem menggunakan *Unified Modeling Language* (UML) guna memperjelas struktur dan alur kerja aplikasi.
4. Untuk menghasilkan aplikasi yang dapat digunakan oleh tiga peran utama—Petugas, Anggota, dan Pengunjung—with antarmuka dan fungsi yang sesuai dengan kebutuhan

masing-masing.

5. Untuk mendukung proses digitalisasi perpustakaan dan meningkatkan kualitas layanan informasi bagi pengguna.

BAB II

PEMBAHASAN

2.1 Tentang Sistem

Sistem Informasi Perpustakaan "Mari Maca" adalah sebuah aplikasi yang dikembangkan menggunakan bahasa pemrograman Java dengan Swing untuk antarmuka pengguna (GUI) dan MySQL sebagai sistem manajemen basis data. Aplikasi ini dirancang untuk mengelola berbagai aspek operasional perpustakaan secara digital, mulai dari manajemen koleksi buku, data anggota, hingga transaksi peminjaman dan pengumuman.

Sistem ini memiliki tiga modul utama yang ditujukan untuk tiga jenis pengguna yang berbeda:

1. Modul Publik (Untuk Pengunjung)

Modul ini adalah gerbang utama bagi siapa saja yang datang ke perpustakaan. Tujuannya adalah untuk mencatat data kunjungan dan menyediakan informasi umum.

- Pencatatan Kunjungan Harian
 - Pengunjung yang datang disambut oleh halaman *Welcome*.
 - Terdapat formulir untuk memasukkan Nama Lengkap dan Nomor Telepon.
 - Data ini disimpan ke dalam tabel *pengunjung* di basis data sebagai catatan kunjungan harian. Fitur ini penting untuk statistik dan pelaporan perpustakaan.
- Dashboard Informasi Publik
 - Setelah mengisi data kunjungan, pengunjung akan diarahkan ke *LibraryDashboard*, sebuah dasbor publik yang informatif.
 - Katalog Buku (Daftar Pustaka): Pengunjung dapat melihat daftar seluruh koleksi buku yang dimiliki perpustakaan. Informasi yang ditampilkan meliputi judul, penulis, penerbit, tipe, lokasi rak, dan yang terpenting, status ketersediaan buku ("Tersedia" atau "Dipinjam").
 - Pencarian Informasi: Terdapat fungsi pencarian universal. Pengunjung dapat mencari buku berdasarkan kata kunci (judul, penulis, dll.) atau mencari pengumuman berdasarkan judulnya.

- Melihat Pengumuman: Pengunjung dapat melihat pengumuman terbaru yang dipublikasikan oleh admin, seperti informasi jam operasional, acara, atau buku baru.

2. Modul Anggota (Untuk Anggota Terdaftar)

Modul ini memberikan akses personal kepada anggota perpustakaan untuk mengelola akun dan interaksi mereka dengan perpustakaan.

- Autentikasi dan Registrasi Akun
 - Login Anggota: Anggota dapat masuk ke sistem menggunakan *username* dan *password* melalui *LoginForm*.
 - Keamanan Kata Sandi: Sistem menggunakan metode *hashing* (SHA-256) untuk mengamankan kata sandi anggota yang disimpan di basis data, sehingga kata sandi asli tidak tersimpan secara langsung.
 - Registrasi Akun Mandiri: Anggota yang datanya sudah didaftarkan oleh admin dapat membuat akun online mereka sendiri melalui *RegisterForm*. Proses registrasi ini memerlukan ID Anggota yang valid, memastikan hanya anggota resmi yang bisa membuat akun.
- Dashboard Anggota Personal
 - Setelah berhasil login, anggota akan disambut di dasbor personal mereka (*DashboardFrame*).
 - Informasi Profil: Anggota dapat melihat detail data diri mereka, termasuk ID Anggota, nama, alamat, email, nomor telepon, dan tanggal pendaftaran.
 - Status Keanggotaan dan Peminjaman: Dasbor ini secara dinamis menampilkan status keanggotaan ("Aktif" atau "Tidak Aktif"). Selain itu, anggota bisa melihat jumlah buku yang sedang mereka pinjam dan tanggal jatuh tempo dari setiap buku tersebut.
- Penjelajahan Koleksi Buku
 - Anggota memiliki akses ke *BukuViewPanel*, sebuah panel canggih untuk melihat koleksi buku.
 - Fitur ini menampilkan seluruh buku dalam format tabel yang mudah dibaca dan dapat diurutkan.
 - Terdapat fungsi pencarian dan filter berdasarkan tipe buku, memudahkan anggota untuk menemukan buku yang spesifik.

- Kolom status buku diberi kode warna untuk membedakan buku yang "Tersedia" dan "Dipinjam" secara visual.

3. Modul Admin (Untuk Staf Perpustakaan)

Modul ini adalah pusat kendali sistem, menyediakan fungsionalitas penuh kepada admin untuk mengelola seluruh sumber daya perpustakaan.

- Manajemen Buku (CRUD)

- Admin dapat melakukan operasi *Create, Read, Update, Delete* (CRUD) pada data buku melalui *BookManagementPanel*.
- Tambah & Edit Buku: Admin dapat menambahkan buku baru atau mengedit data buku yang sudah ada melalui sebuah dialog khusus (*BookDialog*).
- Hapus Buku: Admin dapat menghapus data buku dari sistem.
- Pencarian & Statistik: Admin dapat dengan mudah mencari buku dan melihat statistik ringkas mengenai jumlah total buku, yang tersedia, dipinjam, rusak, atau hilang.

- Manajemen Anggota (CRUD & Cetak Kartu)

- Admin memiliki kontrol penuh atas data anggota melalui *MemberManagementPanel*.
- Tambah, Edit, & Hapus Anggota: Admin dapat mendaftarkan anggota baru, memperbarui informasi mereka, dan menonaktifkan atau menghapus keanggotaan.
- Fitur Unggulan: Cetak Kartu Anggota: Sistem memiliki fitur untuk mencetak kartu anggota secara langsung (*printMemberCard*). Kartu yang dihasilkan memiliki desain profesional yang mencakup:
 - Informasi perpustakaan.
 - Data lengkap anggota (ID, nama, alamat, dll.).
 - Status keanggotaan.
 - Nomor seri unik yang digenerasi otomatis.

- Manajemen Peminjaman dan Pengembalian

- Ini adalah fitur inti dari operasional perpustakaan, dikelola melalui *PeminjamanManagementPanel*.
- Membuat Transaksi Peminjaman: Admin dapat membuat transaksi peminjaman baru melalui *PeminjamanDialog*. Dialog ini secara cerdas

hanya menampilkan buku yang tersedia dan anggota yang aktif, untuk mencegah kesalahan input.

- Memproses Pengembalian: Admin dapat memproses pengembalian buku. Saat sebuah buku dikembalikan, sistem secara otomatis memperbarui status peminjaman menjadi "Dikembalikan" dan mengubah status buku menjadi "Tersedia".
 - Integritas Data dengan Transaksi: Proses peminjaman dan pengembalian dilindungi oleh transaksi database. Artinya, pembaruan data di tabel *peminjaman* dan tabel *buku* harus sama-sama berhasil. Jika salah satu gagal, seluruh operasi akan dibatalkan (*rollback*), sehingga menjaga konsistensi data.
- Manajemen Pengumuman (CRUD)
 - Admin dapat membuat dan mengelola pengumuman melalui *PengumumanManagementPanel*.
 - Setiap pengumuman yang dibuat atau diperbarui oleh admin akan secara otomatis ditampilkan di Dashboard Publik untuk dilihat oleh semua pengunjung dan anggota.

2.2 Konsep Pemrograman Berorientasi Objek (OOP) dalam Sistem Perpustakaan

1. Enkapsulasi (Pembungkusan)

Enkapsulasi adalah konsep membungkus data (variabel) dan metode (fungsi) yang beroperasi pada data tersebut ke dalam satu unit tunggal yang disebut Class. Tujuannya adalah untuk menyembunyikan detail implementasi dari dunia luar dan hanya menyediakan antarmuka (metode publik) untuk berinteraksi dengan data tersebut. Ini dicapai dengan menggunakan *access modifier* seperti *private*.

Contoh Penerapan:

- Kelas Model Data: Kelas seperti *Book.java*, *Member.java*, *Peminjaman.java*, dan *Pengumuman.java* adalah contoh sempurna dari enkapsulasi. Mereka membungkus atribut-atribut yang relevan.

Atribut-atribut ini dideklarasikan sebagai *private* untuk mencegah akses langsung dari luar kelas. Contohnya, di kelas *Book*:

```
private int idBuku;  
private String judulBuku;  
private String author;  
// ... dan seterusnya
```

Untuk mengakses atau mengubah nilai atribut *private* ini, disediakan metode publik yang dikenal sebagai *getter* dan *setter*. Contohnya, di kelas *Book*:

```
public int getIdBuku() {  
    return idBuku;  
}  
  
public void setIdBuku(int idBuku) {  
    this.idBuku = idBuku;  
}
```

- Kelas Panel Manajemen: Kelas seperti *BookManagementPanel.java* juga menerapkan enkapsulasi dengan mendeklarasikan komponen UI (seperti *JTable*, *JButton*) dan objek DAO sebagai *private*. Metode di dalam kelas inilah yang mengelola interaksi antar komponen tersebut, seperti *loadBooks()* atau *deleteSelectedBook()*.

2. Pewarisan (*Inheritance*)

Inheritance adalah mekanisme di mana sebuah kelas baru (disebut *subclass* atau *child class*) dapat mewarisi atribut dan metode dari kelas yang sudah ada (*superclass* atau

parent class). Hal ini memungkinkan penggunaan kembali kode (*code reusability*) dan penciptaan hierarki kelas.

Contoh Penerapan:

- Pewarisan Komponen GUI Swing: Konsep ini sangat jelas terlihat pada semua kelas yang bertanggung jawab atas tampilan antarmuka.

Setiap panel manajemen, seperti *BookManagementPanel*, *MemberManagementPanel*, dan *PeminjamanManagementPanel*, merupakan turunan dari kelas *JPanel* milik Swing. Ini ditulis dengan kata kunci *extends*.

```
public class BookManagementPanel extends JPanel { ... } //
public class MemberManagementPanel extends JPanel { ... } //
public class PeminjamanManagementPanel extends JPanel { ... } //
```

- Dengan mewarisi dari *JPanel*, kelas-kelas ini secara otomatis mendapatkan semua fungsi dasar dari sebuah panel (seperti *setLayout()*, *add()*, *setBackground()*, dll.) dan kemudian menambahkan fungsionalitas spesifik untuk manajemen data masing-masing.

Demikian pula, kelas dialog seperti *BookDialog* dan *MemberDialog* adalah turunan dari *JDialog*.

```
public class BookDialog extends JDialog { ... } //
public class MemberDialog extends JDialog { ... } //
```

- Frame utama aplikasi, *DashboardFrame*, mewarisi sifat dan perilaku dari *JFrame*.

3. Polimorfisme (*Polymorphism*)

Polimorfisme (berarti "banyak bentuk") memungkinkan objek dari kelas yang berbeda untuk merespons pesan (pemanggilan metode) yang sama dengan cara yang berbeda. Di Java, ini sering dicapai melalui *method overriding* (mendefinisikan ulang metode dari *superclass*) dan implementasi *interface*.

Contoh Penerapan:

- Event Handling: Implementasi paling umum dari polimorfisme dalam kode ini adalah melalui *event listener*.

Di kelas *BookManagementPanel*, sebuah *ActionListener* dibuat untuk menangani klik tombol. Metode *actionPerformed(ActionEvent e)* adalah metode yang di-*override* dari *interface ActionListener*.

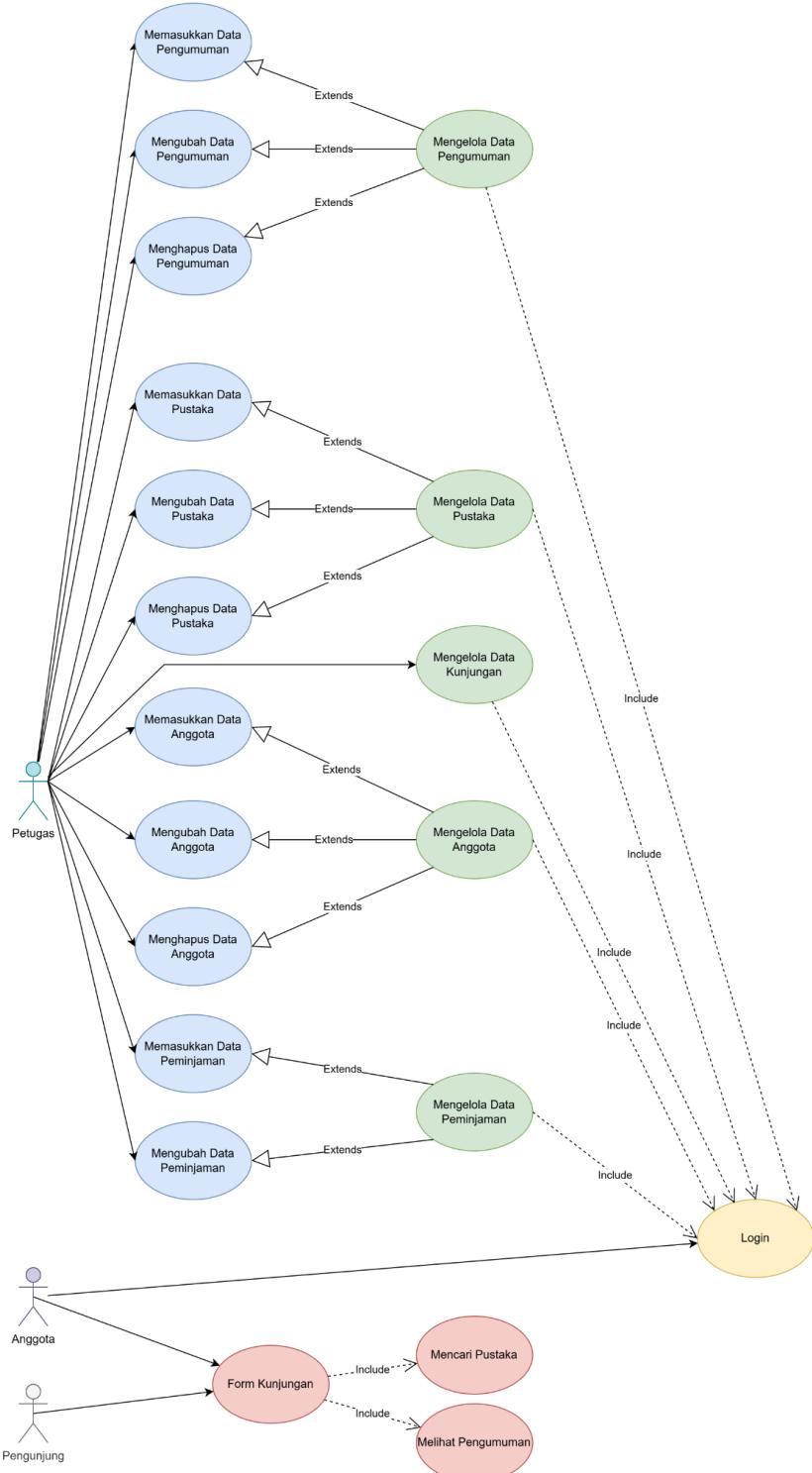
```
addButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        showAddBookDialog();  
    }  
});
```

- Setiap tombol (*addButton, editButton, deleteButton*) memiliki *ActionListener*-nya sendiri, tetapi semuanya merespons "pesan" yang sama (*actionPerformed*) dengan perilaku yang unik (menampilkan dialog tambah, edit, atau hapus).
- *Method Overriding* pada Komponen Swing:
 - Dalam kelas *Welcome.java*, *paintComponent(Graphics g)* di-*override*

untuk menggambar gambar latar belakang pada *JPanel*.

- Pada *MemberManagementPanel.java*, *getTableCellRendererComponent* di-*override* untuk mengubah tampilan sel tabel (misalnya, memberi warna pada status "Aktif" dan "Tidak Aktif").

2.3 Use Case Diagram



Gambar 1 Use Case Diagram

Use Case Diagram digunakan untuk menggambarkan fungsionalitas yang diharapkan dari sistem serta interaksi antara sistem dengan entitas eksternal yang disebut *actor*. Diagram ini memberikan gambaran umum mengenai "siapa" yang dapat melakukan "apa" dalam sistem.

Terdapat tiga *actor* utama dalam sistem ini:

1. Petugas (Administrator): Merupakan pengguna dengan hak akses tertinggi yang bertanggung jawab mengelola seluruh data master dalam sistem.
2. Anggota: Pengguna yang telah terdaftar secara resmi di perpustakaan.
3. Pengunjung: Pengguna umum yang belum menjadi anggota perpustakaan.

Fungsionalitas (Use Case) untuk setiap Actor:

- Petugas
 - Login: Merupakan *use case* yang wajib dilakukan (<<include>>) sebelum dapat mengakses fungsionalitas lainnya.
 - Mengelola Data Pengumuman: *Use case* utama untuk manajemen pengumuman. Ini diperluas (<<extend>>) oleh fungsionalitas spesifik seperti *Memasukkan*, *Mengubah*, dan *Menghapus Data Pengumuman*.
 - Mengelola Data Pustaka: Bertanggung jawab atas manajemen katalog buku, yang mencakup operasi *Memasukkan*, *Mengubah*, dan *Menghapus Data Pustaka*.
 - Mengelola Data Anggota: Mengelola data keanggotaan, termasuk *Memasukkan*, *Mengubah*, dan *Menghapus Data Anggota*.
 - Mengelola Data Peminjaman: Mengelola data transaksi peminjaman buku, yang terdiri dari *Memasukkan* dan *Mengubah Data Peminjaman*.

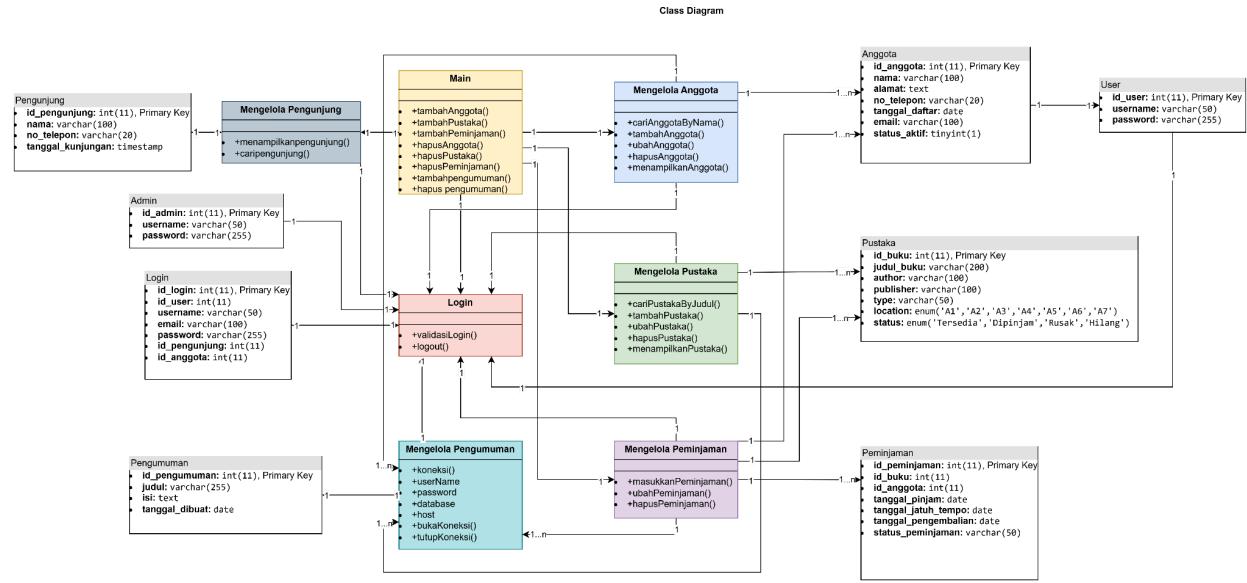
- Anggota dan Pengunjung

Kedua *actor* ini memiliki akses ke fungsionalitas publik.

- Form Kunjungan: Merupakan titik awal interaksi bagi pengunjung dan anggota.
- Mencari Pustaka: Fungsionalitas untuk mencari koleksi buku di perpustakaan. *Use case* ini termasuk (<<include>>) dalam alur setelah mengisi form kunjungan.
- Melihat Pengumuman: Fungsionalitas untuk melihat pengumuman yang telah dipublikasikan oleh petugas. *Use case* ini juga termasuk (<<include>>) dalam

alur setelah mengisi form kunjungan.

2.4 Class Diagram



Gambar 2 Class Diagram

Class Diagram mendeskripsikan struktur statis dari sebuah sistem dengan menunjukkan kelas-kelas sistem, atribut, metode, serta hubungan antar kelas. Diagram pada Gambar 2 menunjukkan komponen-komponen utama yang membangun logika sistem.

Penjelasan Kelas Utama:

- Kelas Entitas (Entity Classes)
 - Merupakan representasi dari objek data yang disimpan dalam basis data.
 - *Pengunjung, Admin, Pengumuman, Pustaka, Anggota, User, dan Peminjaman* adalah kelas entitas.
 - Setiap kelas memiliki atribut yang sesuai dengan kolom pada tabel di basis data, contohnya kelas *Pustaka* memiliki atribut *id_buku*, *judul_buku*, *author*, dan lain-lain.
 - Kelas *User* memiliki relasi dengan *Anggota*, yang menandakan bahwa setiap anggota yang ingin masuk ke sistem harus memiliki akun pengguna.

- Kelas Pengelola (Controller Classes)
 - Kelas-kelas ini bertanggung jawab atas logika bisnis dan pemrosesan data.
 - *Mengelola Pengunjung*, *Mengelola Anggota*, *Mengelola Pustaka*, *Mengelola Peminjaman*, dan *Mengelola Pengumuman* bertindak sebagai *controller*.
 - Contohnya, kelas *Mengelola Pustaka* memiliki metode seperti *cariPustakaByJudul()* dan *tambahPustaka()* yang berinteraksi dengan kelas entitas *Pustaka*.
- Kelas Utama dan Otentikasi
 - *Main*: Berfungsi sebagai kelas orkestrator utama yang mengoordinasikan interaksi antar kelas pengelola.
 - *Login*: Kelas yang secara spesifik menangani proses otentikasi pengguna dengan metode *validasiLogin()* dan *logout()*.

Relasi Antar Kelas:

- Diagram menunjukkan adanya relasi asosiasi antara kelas pengelola dan kelas entitas. Contohnya, *Mengelola Peminjaman* berasosiasi dengan *Peminjaman*.
- Terdapat relasi one-to-many (satu-ke-banyak), misalnya antara *Anggota* dan *Peminjaman* (satu anggota dapat melakukan banyak peminjaman).

2.5 Implementasi Kode PROGRAM

2.5.1 Admin

1. Koneksi Basis Data

```
package com.Library.Admin;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
    public static Connection getConnection() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/db_marimaca_2", "root", ""
            );
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

2. Admin Login Form

```
package com.Library.Admin;

import javax.swing.*;
import java.awt.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class AdminLoginForm extends JFrame {

    private final Color PRIMARY_BLUE = new Color(33, 150, 243);
    private final Color DARK_BLUE = new Color(25, 118, 210);
    private final Color BG_LIGHT = new Color(248, 250, 255);
    private final Color TEXT_COLOR = new Color(50, 50, 50);

    private JTextField usernameField;
    private JPasswordField passwordField;
    private JButton loginButton;
    private JLabel errorLabel;
```

```

public AdminLoginForm() {

    setTitle("Login Admin - Perpustakaan MariMaca");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(400, 300);
    setLocationRelativeTo(null);
    getContentPane().setBackground(BG_LIGHT);
    setLayout(new GridBagLayout());
}

initComponents();
addListeners();
}

private void initComponents() {

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(8, 10, 8, 10);
    gbc.fill = GridBagConstraints.HORIZONTAL;

    // Title
    JLabel titleLabel = new JLabel("LOGIN ADMIN", SwingConstants.CENTER);
    titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 24));
    titleLabel.setForeground(DARK_BLUE);
    gbc.gridx = 0;
}

```

```

gbc.gridx = 0;

gbc.gridwidth = 2;

gbc.insets = new Insets(15, 10, 10, 10);

add(titleLabel, gbc);

gbc.insets = new Insets(8, 10, 8, 10);

// Username Label

JLabel usernameLabel = new JLabel("Username:");

usernameLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));

usernameLabel.setForeground(TEXT_COLOR);

gbc.gridx = 0;

gbc.gridy = 1;

gbc.gridwidth = 1;

add(usernameLabel, gbc);

// Username Field

usernameField = new JTextField(20);

usernameField.setFont(new Font("Segoe UI", Font.PLAIN, 14));

gbc.gridx = 1;

gbc.gridy = 1;

add(usernameField, gbc);

```

```

// Password Label

JLabel passwordLabel = new JLabel("Password:");
passwordLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));
passwordLabel.setForeground(TEXT_COLOR);
gbc.gridx = 0;
gbc.gridy = 2;
add(passwordLabel, gbc);

// Password Field

passwordField = new JPasswordField(20);
passwordField.setFont(new Font("Segoe UI", Font.PLAIN, 14));
gbc.gridx = 1;
gbc.gridy = 2;
add(passwordField, gbc);

// Error Label

errorLabel = new JLabel(" ", SwingConstants.CENTER);
errorLabel.setFont(new Font("Segoe UI", Font.ITALIC, 12));
errorLabel.setForeground(Color.RED);
gbc.gridx = 0;
gbc.gridy = 3;

```

```
gbc.gridx = 2;

gbc.insets = new Insets(0, 10, 0, 10);

add(errorLabel, gbc);

gbc.insets = new Insets(8, 10, 8, 10);

// Login Button

loginButton = new JButton("Login");

loginButton.setFont(new Font("Segoe UI", Font.BOLD, 14));

loginButton.setBackground(PRIMARY_BLUE);

loginButton.setForeground(Color.WHITE);

loginButton.setFocusPainted(false);

loginButton.setPreferredSize(new Dimension(100, 35));

gbc.gridx = 0;

gbc.gridy = 4;

gbc.gridwidth = 2;

gbc.anchor = GridBagConstraints.CENTER;

gbc.fill = GridBagConstraints.NONE;

add(loginButton, gbc);

}

private void addListeners() {
```

```

loginButton.addActionListener(e -> performLogin());

passwordField.addActionListener(e -> performLogin());

}

private void performLogin() {

    String username = usernameField.getText().trim();

    String passwordInput = new String(passwordField.getPassword());

    if (username.isEmpty() || passwordInput.isEmpty()) {

        errorLabel.setText("Username dan password tidak boleh kosong.");

        return;

    }

    errorLabel.setText(" "); // Bersihkan error sebelumnya

}

String sql = "SELECT password, id FROM admin WHERE username = ?";

try (Connection conn = DBConnection.getConnection();

     PreparedStatement pstmt = conn.prepareStatement(sql)) {

    pstmt.setString(1, username);

    ResultSet rs = pstmt.executeQuery();

    if (rs.next()) {

```

```

String storedPassword = rs.getString("password");

// Verifikasi password

if (passwordInput.equals(storedPassword)) {

    JOptionPane.showMessageDialog(this,
        "Login berhasil! Selamat datang Admin.",
        "Sukses",
        JOptionPane.INFORMATION_MESSAGE);

}

dispose(); // Tutup AdminLoginForm

// Buka AdminDashboardFrame

DashboardFrame dashboard = new DashboardFrame();

dashboard.setVisible(true);

} else {

    errorLabel.setText("Username atau password salah.");
}

} else {

    errorLabel.setText("Username atau password salah.");
}

}

} catch (SQLException ex) {

```

```

ex.printStackTrace();

errorLabel.setText("Terjadi kesalahan database. Silakan coba lagi.");

}

}

}

```

3. DashBoardFrame Admin

```

package com.Library.Admin;

import javax.swing.*;
import java.awt.*;

public class DashboardFrame extends JFrame {

    private JButton bookListButton;
    private JButton memberListButton;
    private JButton peminjamanButton;
    private JButton pengumumanButton;
    private JButton pengunjungButton;
    private JButton logoutButton;
    private JButton homeButton;
    private JLabel logoLabel;
    private JPanel mainContentPanel;
    private JPanel homePanel;
}

```

```
private BookManagementPanel bookManagementPanel;  
  
private MemberManagementPanel MemberManagementPanel;  
  
private PeminjamanManagementPanel peminjamanManagementPanel;  
  
private PengumumanManagementPanel pengumumanManagementPanel;  
  
private PengunjungManagementPanel pengunjungManagementPanel;  
  
private final String LOGO_PATH = "image/logo.jpg";  
  
private final Color PRIMARY_GREEN = new Color(67, 160, 71);  
  
private final Color DARK_GREEN = new Color(46, 125, 50);  
  
private final Color SOFT_GREEN = new Color(129, 199, 132);  
  
private final Color BG_LIGHT = new Color(248, 252, 248);  
  
private final Color SIDEBAR_GREEN = new Color(56, 142, 60);  
  
private final Color HOVER_GREEN = new Color(76, 175, 80);  
  
public DashboardFrame() {  
    initComponents();  
    setupLayout();  
    setupEvents();  
    loadSavedLogo();  
    showHomePanel(); // tampilkan panel home secara default  
}
```

```
private void initComponents() {  
    setTitle("Dashboard Admin - Sistem Perpustakaan Mari Maca");  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    setSize(1400, 800);  
    setLocationRelativeTo(null);  
  
    logoLabel = new JLabel();  
  
    homeButton = new JButton("HOME");  
    bookListButton = new JButton("DAFTAR BUKU");  
    memberListButton = new JButton("DAFTAR ANGGOTA");  
    peminjamanButton = new JButton("PEMINJAMAN");  
    pengumumanButton = new JButton("PENGUMUMAN");  
    pengunjungButton = new JButton("PENGUNJUNG");  
    logoutButton = new JButton("KELUAR");  
  
    logoutButton.setBackground(new Color(139, 0, 0));  
    logoutButton.setForeground(Color.WHITE);  
    logoutButton.setFont(new Font("Segoe UI", Font.BOLD, 14));  
    logoutButton.setPreferredSize(new Dimension(280, 50));  
    logoutButton.setMaximumSize(new Dimension(280, 50));
```

```
logoutButton.setFocusPainted(false);

logoutButton.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));

logoutButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

logoutButton.setOpaque(true);

logoutButton.setContentAreaFilled(true);

logoutButton.setHorizontalAlignment(SwingConstants.LEFT);

styleActiveButton(homeButton);

styleSidebarButton(bookListButton);

styleSidebarButton(memberListButton);

styleSidebarButton(peminjamanButton);

styleSidebarButton(pengumumanButton);

styleSidebarButton(pengunjungButton);

createHomePanel();

bookManagementPanel = new BookManagementPanel();

MemberManagementPanel = new MemberManagementPanel();

peminjamanManagementPanel = new PeminjamanManagementPanel();

pengumumanManagementPanel = new PengumumanManagementPanel();

pengunjungManagementPanel = new PengunjungManagementPanel();

mainContentPanel = new JPanel(new CardLayout());
```

```
mainContentPanel.setBackground(BG_LIGHT);

}

private void styleActiveButton(JButton button) {

    button.setBackground(PRIMARY_GREEN);

    button.setForeground(Color.WHITE);

    button.setFont(new Font("Segoe UI", Font.BOLD, 14));

    button.setPreferredSize(new Dimension(280, 50));

    button.setMaximumSize(new Dimension(280, 50));

    button.setFocusPainted(false);

    button.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));

    button.setCursor(new Cursor(Cursor.HAND_CURSOR));

    button.setOpaque(true);

    button.setContentAreaFilled(true);

    button.setHorizontalAlignment(SwingConstants.LEFT);

}

private void styleSidebarButton(JButton button) {

    button.setBackground(SOFT_GREEN);

    button.setForeground(Color.WHITE);

    button.setFont(new Font("Segoe UI", Font.BOLD, 14));

    button.setPreferredSize(new Dimension(280, 50));
```

```
button.setMaximumSize(new Dimension(280, 50));  
  
button.setFocusPainted(false);  
  
button.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));  
  
button.setCursor(new Cursor(Cursor.HAND_CURSOR));  
  
button.setOpaque(true);  
  
button.setContentAreaFilled(true);  
  
button.setHorizontalAlignment(SwingConstants.LEFT);  
  
  
  
  
button.addMouseListener(new java.awt.event.MouseAdapter() {  
  
    public void mouseEntered(java.awt.event.MouseEvent evt) {  
  
        if (button.getBackground().equals(SOFT_GREEN)) {  
  
            button.setBackground(HOVER_GREEN);  
  
        }  
  
    }  
  
    public void mouseExited(java.awt.event.MouseEvent evt) {  
  
        if (button.getBackground().equals(HOVER_GREEN)) {  
  
            button.setBackground(SOFT_GREEN);  
  
        }  
  
    }  
  
});  
  
}
```

```

private void setActiveButton(JButton activeButton) {
    styleSidebarButton(homeButton);
    styleSidebarButton(bookListButton);
    styleSidebarButton(memberListButton);
    styleSidebarButton(peminjamanButton);
    styleSidebarButton(pengumumanButton);
    styleSidebarButton(pengunjungButton);

    styleActiveButton(activeButton);
}

private void createHomePanel() {
    // Inisialisasi panel utama dengan GridBagLayout dan background terang
    homePanel = new JPanel(new GridBagLayout());
    homePanel.setBackground(BG_LIGHT);

    // Setup constraint untuk positioning komponen
    GridBagConstraints gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.anchor = GridBagConstraints.CENTER;
    gbc.insets = new Insets(0, 0, 30, 0);
}

```

```

// Section logo dengan border hijau dan background putih

JPanel logoSection = new JPanel(new GridBagLayout());

logoSection.setBackground(Color.WHITE);

logoSection.setPreferredSize(new Dimension(250, 250));

logoSection.setBorder(BorderFactory.createCompoundBorder(
    BorderFactory.createLineBorder(PRIMARY_GREEN, 3),
    BorderFactory.createEmptyBorder(20, 20, 20, 20)
));

logoSection.add(logoLabel);

// Tambahkan logo section ke panel utama

homePanel.add(logoSection, gbc);

// Tambahkan label welcome pertama - "Selamat Menikmati Layanan Online"

gbc.gridx = 1;

JLabel welcomeLabel1 = new JLabel("Selamat Menikmati Layanan Online",
    SwingConstants.CENTER);

welcomeLabel1.setFont(new Font("Segoe UI", Font.BOLD, 28));

welcomeLabel1.setForeground(DARK_GREEN);

homePanel.add(welcomeLabel1, gbc);

// Tambahkan label welcome kedua - "Perpustakaan Mari Maca"

```

```

gbc.gridx = 2;

JLabel welcomeLabel2 = new JLabel("Perpustakaan Mari Maca", SwingConstants.CENTER);

welcomeLabel2.setFont(new Font("Segoe UI", Font.BOLD, 28));

welcomeLabel2.setForeground(DARK_GREEN);

homePanel.add(welcomeLabel2, gbc);

}

private void setupLayout() {

setLayout(new BorderLayout());

getContentPane().setBackground(BG_LIGHT);

}

add(createSidebar(), BorderLayout.WEST);

mainContentPanel.add(homePanel, "HOME");

mainContentPanel.add(bookManagementPanel, "BOOKS");

mainContentPanel.add(MemberManagementPanel, "MEMBERS");

mainContentPanel.add(peminjamanManagementPanel, "PEMINJAMAN");

mainContentPanel.add(pengumumanManagementPanel, "PENGUMUMAN");

mainContentPanel.add(pengunjungManagementPanel, "PENGUNJUNG");

add(mainContentPanel, BorderLayout.CENTER);

}

```

```
private JPanel createSidebar() {  
  
    // Panel sidebar utama dengan BorderLayout dan background hijau  
  
    JPanel sidebar = new JPanel(new BorderLayout());  
  
    sidebar.setBackground(SIDEBAR_GREEN);  
  
    sidebar.setPreferredSize(new Dimension(320, 0));  
  
  
  
    // Section header untuk menampilkan info user admin  
  
    JPanel userSection = new JPanel(new FlowLayout(FlowLayout.CENTER));  
  
    userSection.setBackground(DARK_GREEN);  
  
    userSection.setPreferredSize(new Dimension(0, 80));  
  
  
  
    // Label untuk menampilkan admin  
  
    JLabel userLabel = new JLabel("Admin");  
  
    userLabel.setFont(new Font("Segoe UI", Font.BOLD, 18));  
  
    userLabel.setForeground(Color.WHITE);  
  
    userLabel.setBorder(BorderFactory.createEmptyBorder(0, 10, 0, 0));  
  
  
  
    userSection.add(userLabel);  
  
  
  
    // Section untuk menu navigasi dengan layout vertical  
  
    JPanel menuSection = new JPanel();  
  
    menuSection.setLayout(new BoxLayout(menuSection, BoxLayout.Y_AXIS));
```

```

menuSection.setBackground(SIDEBAR_GREEN);

menuSection.setBorder(BorderFactory.createEmptyBorder(30, 20, 20, 20));

// Tambahkan semua button menu dengan spacing antar button

menuSection.add(homeButton);

menuSection.add(Box.createRigidArea(new Dimension(0, 10)));

menuSection.add(bookListButton);

menuSection.add(Box.createRigidArea(new Dimension(0, 10)));

menuSection.add(memberListButton);

menuSection.add(Box.createRigidArea(new Dimension(0, 10)));

menuSection.add(peminjamanButton);

menuSection.add(Box.createRigidArea(new Dimension(0, 10)));

menuSection.add(pengumumanButton);

menuSection.add(Box.createRigidArea(new Dimension(0, 10)));

menuSection.add(pengunjungButton);

menuSection.add(Box.createRigidArea(new Dimension(0, 10)));


// Section untuk button logout di bagian bawah

JPanel bottomSection = new JPanel(new FlowLayout(FlowLayout.CENTER));

bottomSection.setBackground(SIDEBAR_GREEN);

bottomSection.setBorder(BorderFactory.createEmptyBorder(20, 20, 30, 20));

bottomSection.add(logoutButton);

```

```
// Section untuk nama perpustakaan di footer

JPanel titleSection = new JPanel();

titleSection.setLayout(new BoxLayout(titleSection, BoxLayout.Y_AXIS));

titleSection.setBackground(SIDEBAR_GREEN);

titleSection.setBorder(BorderFactory.createEmptyBorder(0, 20, 20, 20));

// footer pertama

JLabel titleLabel1 = new JLabel("Layanan Online Perpustakaan");

titleLabel1.setFont(new Font("Segoe UI", Font.BOLD, 12));

titleLabel1.setForeground(Color.WHITE);

titleLabel1.setAlignmentX(Component.CENTER_ALIGNMENT);

// footer kedua

JLabel titleLabel2 = new JLabel("Perpustakaan Mari Maca Kab. Karawang");

titleLabel2.setFont(new Font("Segoe UI", Font.BOLD, 12));

titleLabel2.setForeground(Color.WHITE);

titleLabel2.setAlignmentX(Component.CENTER_ALIGNMENT);

titleSection.add(titleLabel1);

titleSection.add(titleLabel2);
```

```

// Susun layout sidebar: user section di atas, menu di tengah

sidebar.add(userSection, BorderLayout.NORTH);

sidebar.add(menuSection, BorderLayout.CENTER);

// Gabungkan bottom section (logout) dan title section jadi satu panel

JPanel bottomPanel = new JPanel(new BorderLayout());

bottomPanel.setBackground(SIDEAR_GREEN);

bottomPanel.add(bottomSection, BorderLayout.NORTH);

bottomPanel.add(titleSection, BorderLayout.SOUTH);

// Tambahkan bottom panel ke sidebar bagian bawah

sidebar.add(bottomPanel, BorderLayout.SOUTH);

return sidebar;

}

private void loadSavedLogo() {

// Load gambar logo dari path yang sudah ditentukan

ImageIcon icon = new ImageIcon(LOGO_PATH);

// Ambil ukuran asli gambar logo

int width = icon.getIconWidth();

```

```

int height = icon.getIconHeight();

// Tentukan ukuran maksimal untuk logo

int maxWidth = 250;

int maxHeight = 250;

// Cek apakah gambar perlu di-resize karena terlalu besar

if (width > maxWidth || height > maxHeight) {

    // Hitung rasio pengecilan untuk width dan height

    double widthRatio = (double) maxWidth / width;

    double heightRatio = (double) maxHeight / height;

    // Pilih rasio terkecil agar gambar tetap proporsional

    double scale = Math.min(widthRatio, heightRatio);

    // Hitung ukuran baru berdasarkan rasio pengecilan

    int newWidth = (int) (width * scale);

    int newHeight = (int) (height * scale);

    // Resize gambar dengan kualitas smooth dan set ke logoLabel

    Image scaledImage = icon.getImage().getScaledInstance(newWidth, newHeight,
Image.SCALE_SMOOTH);

    logoLabel.setIcon(new ImageIcon(scaledImage));
}

```

```

} else {

    // set gambar asli ke logoLabel

    logoLabel.setIcon(icon);

}

// Set ukuran preferred untuk logoLabel sesuai ukuran final gambar

    logoLabel.setPreferredSize(new Dimension(Math.min(width, maxWidth), Math.min(height,
maxHeight)));

}

private void setupEvents() {

    homeButton.addActionListener(e -> showHomePanel());

    bookListButton.addActionListener(e -> showBookPanel());

    memberListButton.addActionListener(e -> showMemberPanel());

    peminjamanButton.addActionListener(e -> showPeminjamanPanel());

    pengumumanButton.addActionListener(e -> showPengumumanPanel());

    pengunjungButton.addActionListener(e -> showPengunjungPanel());

}

logoutButton.addActionListener(e -> {

    int option = JOptionPane.showConfirmDialog(this,
        "Apakah Anda yakin ingin keluar?",

        "Konfirmasi Keluar",

        JOptionPane.YES_NO_OPTION);

```

```

if (option == JOptionPane.YES_OPTION) {

    this.dispose();

}

});

}

private void showHomePanel() {

    CardLayout cardLayout = (CardLayout) mainContentPanel.getLayout();

    cardLayout.show(mainContentPanel, "HOME");

    setActiveButton(homeButton);

}

private void showBookPanel() {

    CardLayout cardLayout = (CardLayout) mainContentPanel.getLayout();

    cardLayout.show(mainContentPanel, "BOOKS");

    setActiveButton(bookListButton);

}

private void showMemberPanel() {

    CardLayout cardLayout = (CardLayout) mainContentPanel.getLayout();

    cardLayout.show(mainContentPanel, "MEMBERS");

    setActiveButton(memberListButton);
}

```

```
}

private void showPeminjamanPanel() {
    CardLayout cardLayout = (CardLayout) mainContentPanel.getLayout();
    cardLayout.show(mainContentPanel, "PEMINJAMAN");
    setActiveButton(peminjamanButton);
}

private void showPengumumanPanel() {
    CardLayout cardLayout = (CardLayout) mainContentPanel.getLayout();
    cardLayout.show(mainContentPanel, "PENGUMUMAN");
    setActiveButton(pengumumanButton);
}

private void showPengunjungPanel() {
    CardLayout cardLayout = (CardLayout) mainContentPanel.getLayout();
    cardLayout.show(mainContentPanel, "PENGUNJUNG");
    setActiveButton(pengunjungButton);
}

}
```

4. Book

```
package com.Library.Admin;

public class Book {

    private int idBuku;
    private String judulBuku;
    private String author;
    private String publisher;
    private String type;
    private String Location;
    private String status;

    public Book() {}

    public Book(int idBuku, String judulBuku, String author, String publisher, String type, String Location, String status) {
        this.idBuku = idBuku;
        this.judulBuku = judulBuku;
        this.author = author;
        this.publisher = publisher;
        this.type = type;
        this.Location = Location;
        this.status = status;
    }
}
```

```
}

public int getIdBuku() {
    return idBuku;
}

public void setIdBuku(int idBuku) {
    this.idBuku = idBuku;
}

public String getJudulBuku() {
    return judulBuku;
}

public void setJudulBuku(String judulBuku) {
    this.judulBuku = judulBuku;
}

public String getAuthor() {
    return author;
}
```

```
public void setAuthor(String author) {  
    this.author = author;  
}  
  
public String getPublisher() {  
    return publisher;  
}  
  
public void setPublisher(String publisher) {  
    this.publisher = publisher;  
}  
  
public String getType() {  
    return type;  
}  
  
public void setType(String type) {  
    this.type = type;  
}  
  
public String getLocation() {  
    return Location;
```

```
}

public void setLocation(String Location) {
    this.Location = Location;
}

public String getStatus() {
    return status;
}

public void setStatus(String status) {
    this.status = status;
}

@Override

public String toString() {
    return "Book [ID=" + idBuku + ", Judul=" + judulBuku + ", Author=" + author +
           ", Publisher=" + publisher + ", Type=" + type + ", Location=" + Location + ", Status=" +
           status + "]";
}

}
```

5. BookDAO (Data Access Object)

```
package com.Library.Admin;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class BookDAO {

    private Connection connection;

    public BookDAO() {
        this.connection = DBConnection.getConnection();
    }

    public boolean addBook(Book book) {
        String sql = "INSERT INTO buku (judul_buku, author, publisher, type, location, status) VALUES (?, ?, ?, ?, ?, ?)";

        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setString(1, book.getJudulBuku());
            pstmt.setString(2, book.getAuthor());
            pstmt.setString(3, book.getPublisher());
            pstmt.setString(4, book.getType());
            pstmt.setString(5, book.getLocation());
        }
    }
}
```

```

        pstmt.setString(6, book.getStatus());

    }

    int result = pstmt.executeUpdate();

    return result > 0;

} catch (SQLException e) {

    e.printStackTrace();

    return false;

}

}

public List<Book> getAllBooks() {

    List<Book> books = new ArrayList<>();

    String sql = "SELECT * FROM buku ORDER BY id_buku";

    try (Statement stmt = connection.createStatement()) {

        ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {

                Book book = new Book();

                book.setIdBuku(rs.getInt("id_buku"));

                book.setJudulBuku(rs.getString("judul_buku"));

                book.setAuthor(rs.getString("author"));

```

```

        book.setPublisher(rs.getString("publisher"));

        book.setType(rs.getString("type"));

        book.setLocation(rs.getString("location"));

        book.setStatus(rs.getString("status"));

        books.add(book);

    }

} catch (SQLException e) {

    e.printStackTrace();

}

return books;
}

public boolean updateBook(Book book) {

    String sql = "UPDATE buku SET judul_buku=?, author=?, publisher=?, type=?, location=?, status=? WHERE id_buku=?";

    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

        pstmt.setString(1, book.getJudulBuku());

        pstmt.setString(2, book.getAuthor());

        pstmt.setString(3, book.getPublisher());

        pstmt.setString(4, book.getType());

        pstmt.setString(5, book.getLocation());

        pstmt.setString(6, book.getStatus());
    }
}

```

```
pstmt.setInt(7, book.getIdBuku());  
  
int result = pstmt.executeUpdate();  
  
return result > 0;  
} catch (SQLException e) {  
    e.printStackTrace();  
    return false;  
}  
}  
  
public boolean deleteBook(int idBuku) {  
    String sql = "DELETE FROM buku WHERE id_buku=?";  
    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {  
        pstmt.setInt(1, idBuku);  
        int result = pstmt.executeUpdate();  
        return result > 0;  
    } catch (SQLException e) {  
        e.printStackTrace();  
        return false;  
    }  
}
```

```

public List<Book> searchBooks(String keyword) {

    List<Book> books = new ArrayList<>();

    String sql = "SELECT * FROM buku WHERE judul_buku LIKE ? OR author LIKE ? OR Type
LIKE ? OR Location LIKE ? OR Status LIKE ? ORDER BY id_buku";

    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

        String searchPattern = "%" + keyword + "%";
        pstmt.setString(1, searchPattern);
        pstmt.setString(2, searchPattern);
        pstmt.setString(3, searchPattern);
        pstmt.setString(4, searchPattern);
        pstmt.setString(5, searchPattern);

        ResultSet rs = pstmt.executeQuery();

        while (rs.next()) {

            Book book = new Book();

            book.setIdBuku(rs.getInt("id_buku"));

            book.setJudulBuku(rs.getString("judul_buku"));

            book.setAuthor(rs.getString("author"));

            book.setPublisher(rs.getString("publisher"));

            book.setType(rs.getString("type"));

            book.setLocation(rs.getString("location"));

            book.setStatus(rs.getString("status"));
        }
    }
}

```

```

        books.add(book);

    }

} catch (SQLException e) {

    e.printStackTrace();

}

return books;
}
}
```

6. Book Dialog

```

package com.Library.Admin;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class BookDialog extends JDialog {

    private JTextField judulField;
    private JTextField authorField;
    private JTextField publisherField;
```

```
private JTextField typeField;

private JTextField locationField;

private JComboBox<String> statusComboBox;

private JButton saveButton;

private JButton cancelButton;

private boolean confirmed = false;

private Book book;

private final Color PRIMARY_GREEN = new Color(34, 139, 34);

private final Color DARK_GREEN = new Color(0, 100, 0);

private final Color BG_GREEN = new Color(240, 255, 240);

public BookDialog(Frame parent, String title, Book book) {

    super(parent, title, true);

    this.book = book;

    initComponents();

    setupLayout();

    setupEvents();

}

if (book != null) {

    populateFields();

}
```

```
}

private void initComponents() {
    setSize(600, 500);
    setLocationRelativeTo(getParent());
    setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
}

judulField = new JTextField(20);
authorField = new JTextField(20);
publisherField = new JTextField(20);
typeField = new JTextField(20);
locationField = new JTextField(20);

String[] statusOptions = {"Tersedia", "Dipinjam", "Rusak", "Hilang"};
statusComboBox = new JComboBox(statusOptions);

saveButton = new JButton("Simpan");
cancelButton = new JButton("Batal");

styleTextField(judulField);
styleTextField(authorField);
styleTextField(publisherField);
```

```

styleTextField(typeField);

styleTextField(locationField);

styleComboBox(statusComboBox);

styleButton(saveButton, PRIMARY_GREEN);

styleButton(cancelButton, new Color(220, 20, 60));

}

private void styleTextField(JTextField field) {

    field.setFont(new Font("Arial", Font.PLAIN, 14));

    field.setBorder(BorderFactory.createCompoundBorder(
        BorderFactory.createLineBorder(PRIMARY_GREEN, 1),
        BorderFactory.createEmptyBorder(5, 8, 5, 8)
    ));

}

private void styleComboBox(JComboBox<String> comboBox) {

    comboBox.setFont(new Font("Arial", Font.PLAIN, 14));

    comboBox.setBackground(Color.WHITE);

    comboBox.setBorder(BorderFactory.createLineBorder(PRIMARY_GREEN, 1));

}

private void styleButton(JButton button, Color color) {

```

```
button.setBackground(color);

button.setForeground(Color.WHITE);

button.setFont(new Font("Arial", Font.BOLD, 14));

button.setPreferredSize(new Dimension(120, 40));

button.setFocusPainted(false);

button.setCursor(new Cursor(Cursor.HAND_CURSOR));

button.setBorder(BorderFactory.createRaisedBevelBorder());

}

private void setupLayout() {

setLayout(new BorderLayout());

JPanel headerPanel = new JPanel();

headerPanel.setBackground(DARK_GREEN);

headerPanel.setPreferredSize(new Dimension(0, 60));

JLabel titleLabel = new JLabel(getTitle());

titleLabel.setFont(new Font("Arial", Font.BOLD, 18));

titleLabel.setForeground(Color.WHITE);

titleLabel.setHorizontalAlignment(SwingConstants.CENTER);

headerPanel.add(titleLabel);
```

```

add(headerPanel, BorderLayout.NORTH);

JPanel formPanel = new JPanel(new GridBagLayout());
formPanel.setBackground(BG_GREEN);
formPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(8, 8, 8, 8);
gbc.anchor = GridBagConstraints.WEST;

addFormField(formPanel, gbc, 0, "Judul Buku:", judulField);
addFormField(formPanel, gbc, 1, "Author:", authorField);
addFormField(formPanel, gbc, 2, "Publisher:", publisherField);
addFormField(formPanel, gbc, 3, "Type:", typeField);
addFormField(formPanel, gbc, 4, "Location:", locationField);
addFormField(formPanel, gbc, 5, "Status:", statusComboBox);

add(formPanel, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 20));
buttonPanel.setBackground(BG_GREEN);
buttonPanel.add(saveButton);

```

```
buttonPanel.add(cancelButton);

add(buttonPanel, BorderLayout.SOUTH);

}

private void addFormField(JPanel panel, GridBagConstraints gbc, int row, String labelText,
JComponent field) {

JLabel label = new JLabel(labelText);

label.setFont(new Font("Arial", Font.BOLD, 14));

label.setForeground(DARK_GREEN);

gbc.gridx = 0;

gbc.gridy = row;

gbc.gridwidth = 1;

panel.add(label, gbc);

gbc.gridx = 1;

gbc.gridy = row;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.HORIZONTAL;

panel.add(field, gbc);

gbc.fill = GridBagConstraints.NONE;

}
```

```
private void setupEvents() {  
  
    saveButton.addActionListener(new ActionListener() {  
  
        @Override  
  
        public void actionPerformed(ActionEvent e) {  
  
            saveBook();  
  
        }  
  
    });  
  
    cancelButton.addActionListener(new ActionListener() {  
  
        @Override  
  
        public void actionPerformed(ActionEvent e) {  
  
            dispose();  
  
        }  
  
    });  
  
}  
  
private void populateFields() {  
  
    if (book != null) {  
  
        judulField.setText(book.getJudulBuku());  
  
        authorField.setText(book.getAuthor());  
  
        publisherField.setText(book.getPublisher());  
    }  
}
```



```
return;  
}  
  
if (typeField.getText().trim().isEmpty()) {  
  
    showError("Type tidak boleh kosong!");  
  
    typeField.requestFocus();  
  
    return;  
}  
  
  
if (locationField.getText().trim().isEmpty()) {  
  
    showError("Location tidak boleh kosong!");  
  
    locationField.requestFocus();  
  
    return;  
}  
  
  
if (book == null) {  
  
    book = new Book();  
  
}  
  
  
book.setJudulBuku(judulField.getText().trim());  
  
book.setAuthor(authorField.getText().trim());  
  
book.setPublisher(publisherField.getText().trim());
```

```

book.setType(typeField.getText().trim());

book.setLocation(locationField.getText().trim());

book.setStatus((String) statusComboBox.getSelectedItem());

confirmed = true;

dispose();

}

private void showError(String message) {

JOptionPane.showMessageDialog(this,
    "Warning!" + message,
    "Input Error",
    JOptionPane.WARNING_MESSAGE);

}

public boolean isConfirmed() {

    return confirmed;

}

public Book getBook() {

    return book;

}

```

```
}
```

7. Book Management Panel

```
package com.Library.Admin;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.DefaultTableCellRenderer;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

public class BookManagementPanel extends JPanel {

    private BookDAO bookDAO;
    private JTable bookTable;
    private DefaultTableModel tableModel;
    private JTextField searchField;
    private JButton addButton, editButton, deleteButton, refreshButton, searchButton;
    private JLabel statusLabel;
```

```

// Warna hijau theme

private final Color PRIMARY_GREEN = new Color(34, 139, 34); // Forest Green

private final Color LIGHT_GREEN = new Color(144, 238, 144); // Light Green

private final Color DARK_GREEN = new Color(0, 100, 0); // Dark Green

private final Color ACCENT_GREEN = new Color(50, 205, 50); // Lime Green

private final Color BG_GREEN = new Color(240, 255, 240); // Honeydew

public BookManagementPanel() {

    bookDAO = new BookDAO();

    initComponents();

    setupLayout();

    setupEvents();

    loadBooks();

}

private void initComponents() {

    setBackground(Color.WHITE);

}

// Table

String[] columns = {"ID", "Judul Buku", "Author", "Publisher", "Type", "Location", "Status"};

tableModel = new DefaultTableModel(columns, 0) {

```

```

@Override

public boolean isCellEditable(int row, int column) {

    return false; // Tabel tidak bisa diedit langsung

};

bookTable = new JTable(tableModel);

bookTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

bookTable.setRowHeight(30);

bookTable.setFont(new Font("Arial", Font.PLAIN, 12));

bookTable.getTableHeader().setFont(new Font("Arial", Font.BOLD, 13));

bookTable.getTableHeader().setBackground(PRIMARY_GREEN);

bookTable.getTableHeader().setForeground(Color.WHITE);

bookTable.setGridColor(LIGHT_GREEN);

bookTable.setSelectionBackground(LIGHT_GREEN);

bookTable.setSelectionForeground(DARK_GREEN);

// Mengatur lebar kolom

bookTable.getColumnModel().getColumn(0).setPreferredWidth(50); // ID

bookTable.getColumnModel().getColumn(1).setPreferredWidth(200); // Judul

bookTable.getColumnModel().getColumn(2).setPreferredWidth(150); // Author

bookTable.getColumnModel().getColumn(3).setPreferredWidth(150); // Publisher

bookTable.getColumnModel().getColumn(4).setPreferredWidth(100); // Type

```

```

bookTable.getColumnModel().getColumn(5).setPreferredWidth(100); // Location

bookTable.getColumnModel().getColumn(6).setPreferredWidth(100); // Status

// Custom renderer untuk kolom Status

bookTable.getColumnModel().getColumn(6).setCellRenderer(new DefaultTableCellRenderer() {

    @Override

    public Component getTableCellRendererComponent(JTable table, Object value,
        boolean isSelected, boolean hasFocus, int row, int column) {

        Component c = super.getTableCellRendererComponent(table, value, isSelected, hasFocus,
            row, column);

        if (!isSelected) {

            if ("Tersedia".equals(value)) {

                c.setBackground(new Color(220, 255, 220));

                c.setForeground(DARK_GREEN);

            } else if ("Dipinjam".equals(value)) {

                c.setBackground(new Color(230, 240, 255));

                c.setForeground(new Color(0, 70, 140));

            } else if ("Rusak".equals(value)) {

                c.setBackground(new Color(255, 200, 200));

                c.setForeground(new Color(32, 0, 0));

            }

        } else {

    }
}

```

```

        c.setBackground(Color.WHITE);

        c.setForeground(Color.BLACK);

    }

}

setHorizontalAlignment(SwingConstants.CENTER);

return c;

}

});

// Search field

searchField = new JTextField(20);

searchField.setFont(new Font("Arial", Font.PLAIN, 14));

// Status label

statusLabel = new JLabel("Total: 0 buku");

statusLabel.setFont(new Font("Arial", Font.BOLD, 12));

statusLabel.setForeground(DARK_GREEN);

// Buttons

 addButton = new JButton("Tambah Buku");

 editButton = new JButton("Edit Buku");

 deleteButton = new JButton("Hapus Buku");

```

```

refreshButton = new JButton("Refresh");

searchButton = new JButton("Cari");

// Style buttons

styleButton(addButton, ACCENT_GREEN);

styleButton(editButton, new Color(255, 165, 0));

styleButton(deleteButton, new Color(139, 0, 0));

styleButton(refreshButton, PRIMARY_GREEN);

styleButton(searchButton, DARK_GREEN);

}

private void styleButton(JButton button, Color color) {

button.setBackground(color);

button.setForeground(Color.WHITE);

button.setFont(new Font("Arial", Font.BOLD, 12));

button.setPreferredSize(new Dimension(130, 35));

button.setFocusPainted(false);

button.setCursor(new Cursor(Cursor.HAND_CURSOR));

button.setBorder(BorderFactory.createRaisedBevelBorder());

}

private void setupLayout() {

```

```
setLayout(new BorderLayout());  
  
// Header Panel  
  
JPanel headerPanel = new JPanel(new BorderLayout());  
  
headerPanel.setBackground(DARK_GREEN);  
  
headerPanel.setPreferredSize(new Dimension(0, 60));  
  
JLabel titleLabel = new JLabel("DAFTAR BUKU PERPUSTAKAAN");  
  
titleLabel.setFont(new Font("Arial", Font.BOLD, 20));  
  
titleLabel.setForeground(Color.WHITE);  
  
titleLabel.setHorizontalAlignment(SwingConstants.CENTER);  
  
headerPanel.add(titleLabel, BorderLayout.CENTER);  
  
add(headerPanel, BorderLayout.NORTH);  
  
// Top Panel (Search + Status)  
  
JPanel topPanel = new JPanel(new BorderLayout());  
  
topPanel.setBackground(BG_GREEN);  
  
topPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));  
  
// Search Panel  
  
JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
```

```
searchPanel.setBackground(BG_GREEN);

JLabel searchLabel = new JLabel("Cari Buku:");
searchLabel.setFont(new Font("Arial", Font.BOLD, 14));
searchLabel.setForeground(DARK_GREEN);
searchPanel.add(searchLabel);

searchPanel.add(Box.createHorizontalStrut(10));
searchPanel.add(searchField);
searchPanel.add(Box.createHorizontalStrut(10));
searchPanel.add(searchButton);

// Status Panel
JPanel statusPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
statusPanel.setBackground(BG_GREEN);
statusPanel.add(statusLabel);

topPanel.add(searchPanel, BorderLayout.WEST);
topPanel.add(statusPanel, BorderLayout.EAST);

add(topPanel, BorderLayout.NORTH);
```

```

// Table Panel

JScrollPane scrollPane = new JScrollPane(bookTable);

scrollPane.setBorder(BorderFactory.createTitledBorder(
    BorderFactory.createLineBorder(PRIMARY_GREEN, 2),
    "Daftar Buku", 0, 0,
    new Font("Arial", Font.BOLD, 14), PRIMARY_GREEN));
scrollPane.setViewport().setBackground(Color.WHITE);

add(scrollPane, BorderLayout.CENTER);

// Button Panel

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 15, 15));

buttonPanel.setBackground(BG_GREEN);
buttonPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

buttonPanel.add(addButton);
buttonPanel.add(editButton);
buttonPanel.add(deleteButton);
buttonPanel.add(refreshButton);

add(buttonPanel, BorderLayout.SOUTH);

}

```

```
private void setupEvents() {  
    addButton.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            showAddBookDialog();  
        }  
    });  
  
    editButton.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            editSelectedBook();  
        }  
    });  
  
    deleteButton.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            deleteSelectedBook();  
        }  
    });
```

```
refreshButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        loadBooks();  
    }  
});  
  
searchButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        searchBooks();  
    }  
});  
  
// Enter key untuk search  
searchField.addActionListener(e -> searchBooks());  
}  
  
private void loadBooks() {  
    SwingUtilities.invokeLater(() -> {  
        try {
```

```

tableModel.setRowCount(0); // Clear table

List<Book> books = bookDAO.getAllBooks();

int tersedia = 0, dipinjam = 0, rusak = 0, hilang = 0;

for (Book book : books) {

    Object[] row = {

        book.getIdBuku(),
        book.getJudulBuku(),
        book.getAuthor(),
        book.getPublisher(),
        book.getType(),
        book.getLocation(),
        book.getStatus()
    };

    tableModel.addRow(row);
}

// Hitung jumlah berdasarkan status

switch (book.getStatus().toLowerCase()) {

    case "tersedia":

        tersedia++; break;

    case "dipinjam":

```

```

        dipinjam++; break;

    case "rusak":

        rusak++; break;

    case "hilang":

        hilang++; break;

    }

}

// Update status info lengkap

int total = books.size();

statusLabel.setText(String.format(
    "<html>Total: %d buku<br/>Tersedia: %d, Dipinjam: %d, <br/>Rusak: %d, Hilang: %d</html>",
    total, tersedia, dipinjam, rusak, hilang
));

}

} catch (Exception e) {

    JOptionPane.showMessageDialog(this,
        "Error loading books: " + e.getMessage(),
        "Error", JOptionPane.ERROR_MESSAGE);

    e.printStackTrace();

}

});

```

```
}
```

```
private void searchBooks() {
```

```
    String keyword = searchField.getText().trim();
```

```
    if (keyword.isEmpty()) {
```

```
        loadBooks();
```

```
        return;
```

```
}
```

```
    SwingUtilities.invokeLater(() -> {
```

```
        try {
```

```
            tableModel.setRowCount(0);
```

```
            List<Book> books = bookDAO.searchBooks(keyword);
```

```
            for (Book book : books) {
```

```
                Object[] row = {
```

```
                    book.getIdBuku(),
```

```
                    book.getJudulBuku(),
```

```
                    book.getAuthor(),
```

```
                    book.getPublisher(),
```

```
                    book.getType(),
```

```

        book.getLocation(),
        book.getStatus()
    );
    tableModel.addRow(row);
}

statusLabel.setText("Hasil Pencarian: " + books.size() + " buku ditemukan");

} catch (Exception e) {
    JOptionPane.showMessageDialog(this,
        "Error searching books: " + e.getMessage(),
        "Error", JOptionPane.ERROR_MESSAGE);
}

});

}

private void showAddBookDialog() {
    // Cari parent frame untuk dialog
    Window parentWindow = SwingUtilities.getWindowAncestor(this);
    Frame parentFrame = (parentWindow instanceof Frame) ? (Frame) parentWindow : null;

    BookDialog dialog = new BookDialog(parentFrame, "Tambah Buku Baru", null);
    dialog.setVisible(true);
}

```

```

if (dialog.isConfirmed()) {

    Book newBook = dialog.getBook();

    if (bookDAO.addBook(newBook)) {

        JOptionPane.showMessageDialog(this,
            "Buku berhasil ditambahkan!",
            "Sukses", JOptionPane.INFORMATION_MESSAGE);

        loadBooks();
    } else {

        JOptionPane.showMessageDialog(this,
            "Gagal menambahkan buku!",
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}

private void editSelectedBook() {

    int selectedRow = bookTable.getSelectedRow();

    if (selectedRow == -1) {

        JOptionPane.showMessageDialog(this,
            "Warning: Pilih buku yang ingin diedit!",
            "Peringatan", JOptionPane.WARNING_MESSAGE);
    }
}

```

```
        return;  
    }  
  
    // Get selected book data  
  
    int idBuku = (Integer) tableModel.getValueAt(selectedRow, 0);  
  
    String judulBuku = (String) tableModel.getValueAt(selectedRow, 1);  
  
    String author = (String) tableModel.getValueAt(selectedRow, 2);  
  
    String publisher = (String) tableModel.getValueAt(selectedRow, 3);  
  
    String Type = (String) tableModel.getValueAt(selectedRow, 4);  
  
    String Location = (String) tableModel.getValueAt(selectedRow, 5);  
  
    String status = (String) tableModel.getValueAt(selectedRow, 6);  
  
    Book selectedBook = new Book(idBuku, judulBuku, author, publisher, Type, Location, status);  
  
    // Cari parent frame untuk dialog  
  
    Window parentWindow = SwingUtilities.getWindowAncestor(this);  
  
    Frame parentFrame = (parentWindow instanceof Frame) ? (Frame) parentWindow : null;  
  
    BookDialog dialog = new BookDialog(parentFrame, "Edit Buku", selectedBook);  
  
    dialog.setVisible(true);  
  
    if (dialog.isConfirmed()) {
```

```

Book updatedBook = dialog.getBook();

updatedBook.setIdBuku(idBuku);

if (bookDAO.updateBook(updatedBook)) {

    JOptionPane.showMessageDialog(this,
        "Buku berhasil diupdate!",
        "Sukses", JOptionPane.INFORMATION_MESSAGE);

    loadBooks();
} else {

    JOptionPane.showMessageDialog(this,
        "Gagal mengupdate buku!",
        "Error", JOptionPane.ERROR_MESSAGE);
}

}

}

private void deleteSelectedBook() {

    int selectedRow = bookTable.getSelectedRow();

    if (selectedRow == -1) {

        JOptionPane.showMessageDialog(this,
            "Warning: Pilih buku yang ingin dihapus!",
            "Peringatan", JOptionPane.WARNING_MESSAGE);
}

```

```

        return;
    }

    int idBuku = (Integer) tableModel.getValueAt(selectedRow, 0);
    String judulBuku = (String) tableModel.getValueAt(selectedRow, 1);

    int option = JOptionPane.showConfirmDialog(this,
        "Yakin ingin menghapus buku:\n" + judulBuku + "?",
        "Konfirmasi Hapus", JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE);

    if (option == JOptionPane.YES_OPTION) {
        if (bookDAO.deleteBook(idBuku)) {
            JOptionPane.showMessageDialog(this,
                "Buku berhasil dihapus!",
                "Sukses", JOptionPane.INFORMATION_MESSAGE);
            loadBooks();
        } else {
            JOptionPane.showMessageDialog(this,
                "Gagal menghapus buku!",
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

```
    }  
  
}  
  
{
```

8. Member

```
package com.Library.Admin;  
  
import java.sql.Date;  
  
public class Member {  
  
    private int idAnggota;  
  
    private String nama;  
  
    private String alamat;  
  
    private String noTelepon;  
  
    private Date tanggalDaftar;  
  
    private String email;  
  
    private String statusAktif;  
  
    public Member() {}  
  
    public Member(int idAnggota, String nama, String alamat, String noTelepon,  
        String email, Date tanggalDaftar, String statusAktif) {  
        this.idAnggota = idAnggota;  
        this.nama = nama;  
        this.alamat = alamat;  
        this.noTelepon = noTelepon;  
        this.email = email;  
        this.tanggalDaftar = tanggalDaftar;  
        this.statusAktif = statusAktif;  
    }  
}
```

```
Date tanggalDaftar, String email, String statusAktif) {  
    this.idAnggota = idAnggota;  
    this.nama = nama;  
    this.alamat = alamat;  
    this.noTelepon = noTelepon;  
    this.tanggalDaftar = tanggalDaftar;  
    this.email = email;  
    this.statusAktif = statusAktif;  
}  
  
public int getIdAnggota() {  
    return idAnggota;  
}  
  
public void setIdAnggota(int idAnggota) {  
    this.idAnggota = idAnggota;  
}  
  
public String getNama() {  
    return nama;  
}
```

```
public void setNama(String nama) {  
    this.nama = nama;  
}  
  
public String getAlamat() {  
    return alamat;  
}  
  
public void setAlamat(String alamat) {  
    this.alamat = alamat;  
}  
  
public String getNoTelepon() {  
    return noTelepon;  
}  
  
public void setNoTelepon(String noTelepon) {  
    this.noTelepon = noTelepon;  
}  
  
public Date getTanggalDaftar() {  
    return tanggalDaftar;
```

```
}

public void setTanggalDaftar(Date tanggalDaftar) {
    this.tanggalDaftar = tanggalDaftar;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getStatusAktif() {
    return statusAktif;
}

public void setStatusAktif(String statusAktif) {
    this.statusAktif = statusAktif;
}
```

```

@Override

public String toString() {

    return "Member [ID=" + idAnggota + ", Nama=" + nama + ", Alamat=" + alamat +
           ", No_Telepon=" + noTelepon + ", Tanggal_Daftar=" + tanggalDaftar +
           ", Email=" + email + ", Status=" + statusAktif + "]";

}

}

```

9. Member DAO (Data Access Object)

```

package com.Library.Admin;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class MemberDAO {

    private Connection connection;

    public MemberDAO() {

        this.connection = DBConnection.getConnection();

    }
}

```

```

public boolean addMember(Member member) {

    String sql = "INSERT INTO anggota (nama, alamat, no_telepon, tanggal_daftar, email,
status_aktif) VALUES (?, ?, ?, ?, ?, ?)";

    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

        pstmt.setString(1, member.getNama());
        pstmt.setString(2, member.getAlamat());
        pstmt.setString(3, member.getNoTelepon());
        pstmt.setDate(4, member.getTanggalDaftar());
        pstmt.setString(5, member.getEmail());
        pstmt.setString(6, "1");

        int result = pstmt.executeUpdate();

        return result > 0;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}

public List<Member> getAllMembers() {
    List<Member> members = new ArrayList<>();

    String sql = "SELECT * FROM anggota ORDER BY id_anggota";

```

```

try (Statement stmt = connection.createStatement());

ResultSet rs = stmt.executeQuery(sql)) {

while (rs.next()) {

Member member = new Member();

member.setIdAnggota(rs.getInt("id_anggota"));

member.setNama(rs.getString("nama"));

member.setAlamat(rs.getString("alamat"));

member.setNoTelepon(rs.getString("no_telepon"));

member.setTanggalDaftar(rs.getDate("tanggal_daftar"));

member.setEmail(rs.getString("email"));

member.setStatusAktif(rs.getString("status_aktif"));

members.add(member);

}

} catch (SQLException e) {

e.printStackTrace();

}

return members;

}

public boolean updateMember(Member member) {

```

```

String sql = "UPDATE anggota SET nama=?, alamat=?, no_telepon=?, tanggal_daftar=?, email=? , status_aktif=? WHERE id_anggota=?";

try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

    pstmt.setString(1, member.getNama());
    pstmt.setString(2, member.getAlamat());
    pstmt.setString(3, member.getNoTelepon());
    pstmt.setDate(4, member.getTanggalDaftar());
    pstmt.setString(5, member.getEmail());
    pstmt.setString(6, member.getStatusAktif());
    pstmt.setInt(7, member.getIdAnggota());

    int result = pstmt.executeUpdate();

    return result > 0;
} catch (SQLException e) {
    e.printStackTrace();
    return false;
}
}

public boolean deleteMember(int idAnggota) {

    String sql = "DELETE FROM anggota WHERE id_anggota=?";

    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

        pstmt.setInt(1, idAnggota);
    }
}

```

```

int result = pstmt.executeUpdate();

return result > 0;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public List<Member> searchMembers(String keyword) {

List<Member> members = new ArrayList<>();

String sql = "SELECT * FROM anggota WHERE nama LIKE ? OR no_telepon LIKE ? OR
status_aktif LIKE ? OR Alamat LIKE ? ORDER BY id_anggota";

try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

String searchPattern = "%" + keyword + "%";

pstmt.setString(1, searchPattern);

pstmt.setString(2, searchPattern);

pstmt.setString(3, searchPattern);

pstmt.setString(4, searchPattern);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

```

```

Member member = new Member();

member.setIdAnggota(rs.getInt("id_anggota"));

member.setNama(rs.getString("nama"));

member.setAlamat(rs.getString("alamat"));

member.setNoTelepon(rs.getString("no_telepon"));

member.setTanggalDaftar(rs.getDate("tanggal_daftar"));

member.setEmail(rs.getString("email"));

member.setStatusAktif(rs.getString("status_aktif"));

members.add(member);

}

} catch (SQLException e) {

e.printStackTrace();

}

return members;

}

public int getActiveMembers() {

String sql = "SELECT COUNT(*) as total FROM anggota WHERE status_aktif = '1'";

try (Statement stmt = connection.createStatement());

ResultSet rs = stmt.executeQuery(sql)) {

if (rs.next()) {

```

```

        return rs.getInt("total");

    }

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

}

return 0;
}

}

public int getInactiveMembers() {

String sql = "SELECT COUNT(*) as total FROM anggota WHERE status_aktif = '0'";

try (Statement stmt = connection.createStatement()) {

    ResultSet rs = stmt.executeQuery(sql)) {

        if (rs.next()) {

            return rs.getInt("total");
        }

    }

} catch (SQLException e) {

    e.printStackTrace();
}

}

return 0;
}

}

```

10. Member Dialog

```
package com.Library.Admin;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Date;
import java.time.LocalDate;

public class MemberDialog extends JDialog {

    private JTextField namaField;
    private JTextArea alamatArea;
    private JTextField noTeleponField;
    private JTextField emailField;
    private JComboBox<String> statusComboBox;
    private JButton saveButton;
    private JButton cancelButton;
    private boolean confirmed = false;
    private Member member;

    private final Color PRIMARY_GREEN = new Color(34, 139, 34);
```

```
private final Color DARK_GREEN = new Color(0, 100, 0);

private final Color BG_GREEN = new Color(240, 255, 240);

public MemberDialog(Frame parent, String title, Member member) {

    super(parent, title, true);

    this.member = member;

    initComponents();

    setupLayout();

    setupEvents();

}

if (member != null) {

    populateFields();

}

}

private void initComponents() {

    setSize(600, 550);

    setLocationRelativeTo(getParent());

    setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);

}

namaField = new JTextField(20);

alamatArea = new JTextArea(3, 20);
```

```

alamatArea.setLineWrap(true);

alamatArea.setWrapStyleWord(true);

noTeleponField = new JTextField(20);

emailField = new JTextField(20);

String[] statusOptions = {"1", "0"}; // 1 = Aktif, 0 = Tidak Aktif

statusComboBox = new JComboBox<>(statusOptions);

saveButton = new JButton("Simpan");

cancelButton = new JButton("Batal");

styleTextField(nomeField);

styleTextArea(alamatArea);

styleTextField(noTeleponField);

styleTextField(emailField);

styleComboBox(statusComboBox);

styleButton(saveButton, PRIMARY_GREEN);

styleButton(cancelButton, new Color(220, 20, 60));

}

private void styleTextField(JTextField field) {

    field.setFont(new Font("Arial", Font.PLAIN, 14));
}

```

```

field.setBorder(BorderFactory.createCompoundBorder(
    BorderFactory.createLineBorder(PRIMARY_GREEN, 1),
    BorderFactory.createEmptyBorder(5, 8, 5, 8)
));
}

private void styleTextArea(JTextArea area) {
    area.setFont(new Font("Arial", Font.PLAIN, 14));
    area.setBorder(BorderFactory.createCompoundBorder(
        BorderFactory.createLineBorder(PRIMARY_GREEN, 1),
        BorderFactory.createEmptyBorder(5, 8, 5, 8)
    ));
}

private void styleComboBox(JComboBox<String> comboBox) {
    comboBox.setFont(new Font("Arial", Font.PLAIN, 14));
    comboBox.setBackground(Color.WHITE);
    comboBox.setBorder(BorderFactory.createLineBorder(PRIMARY_GREEN, 1));
}

// Buat renderer kustom untuk menampilkan teks yang lebih jelas
comboBox.setRenderer(new DefaultListCellRenderer() {
    @Override

```

```

public Component getListCellRendererComponent(JList<?> list, Object value,
                                              int index, boolean isSelected, boolean cellHasFocus) {
    super.getListCellRendererComponent(list, value, index, isSelected, cellHasFocus);
    if ("1".equals(value)) {
        setText("Aktif");
    } else if ("0".equals(value)) {
        setText("Tidak Aktif");
    }
    return this;
}
});

}

private void styleButton(JButton button, Color color) {
    button.setBackground(color);
    button.setForeground(Color.WHITE);
    button.setFont(new Font("Arial", Font.BOLD, 14));
    button.setPreferredSize(new Dimension(120, 40));
    button.setFocusPainted(false);
    button.setCursor(new Cursor(Cursor.HAND_CURSOR));
    button.setBorder(BorderFactory.createRaisedBevelBorder());
}
}

```

```
private void setupLayout() {  
    setLayout(new BorderLayout());  
  
    JPanel headerPanel = new JPanel();  
    headerPanel.setBackground(DARK_GREEN);  
    headerPanel.setPreferredSize(new Dimension(0, 60));  
  
    JLabel titleLabel = new JLabel(getTitle());  
    titleLabel.setFont(new Font("Arial", Font.BOLD, 18));  
    titleLabel.setForeground(Color.WHITE);  
    titleLabel.setHorizontalAlignment(SwingConstants.CENTER);  
    headerPanel.add(titleLabel);  
  
    add(headerPanel, BorderLayout.NORTH);  
  
    JPanel formPanel = new JPanel(new GridBagLayout());  
    formPanel.setBackground(BG_GREEN);  
    formPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));  
  
    GridBagConstraints gbc = new GridBagConstraints();  
    gbc.insets = new Insets(8, 8, 8, 8);
```

```

gbc.anchor = GridBagConstraints.WEST;

addFormField(formPanel, gbc, 0, "Nama Lengkap:", namaField);

// Alamat dengan JScrollPane
JScrollPane alamatScroll = new JScrollPane(alamatArea);
alamatScroll.setPreferredSize(new Dimension(300, 80));
addFormField(formPanel, gbc, 1, "Alamat:", alamatScroll);

addFormField(formPanel, gbc, 2, "No. Telepon:", noTeleponField);
addFormField(formPanel, gbc, 3, "Email:", emailField);

// Status hanya jika member tidak null (edit)
if(member != null) {
    addFormField(formPanel, gbc, 4, "Status:", statusComboBox);
}

// Tambahan info tanggal daftar jika edit
if(member != null) {
    JLabel tanggalInfo = new JLabel("Tanggal Daftar: " + member.getTanggalDaftar());
    tanggalInfo.setFont(new Font("Arial", Font.ITALIC, 12));
    tanggalInfo.setForeground(DARK_GREEN);
    addFormField(formPanel, gbc, 5, "", tanggalInfo);
}

```

```
}

add(formPanel, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 20));

buttonPanel.setBackground(BG_GREEN);

buttonPanel.add(saveButton);

buttonPanel.add(cancelButton);

add(buttonPanel, BorderLayout.SOUTH);

}

private void addFormField(JPanel panel, GridBagConstraints gbc, int row, String labelText,
JComponent field) {

if (!labelText.isEmpty()) {

JLabel label = new JLabel(labelText);

label.setFont(new Font("Arial", Font.BOLD, 14));

label.setForeground(DARK_GREEN);

gbc.gridx = 0;

gbc.gridy = row;

gbc.gridwidth = 1;

panel.add(label, gbc);
```

```

    }

gbc.gridx = labelText.isEmpty() ? 0 : 1;
gbc.gridy = row;
gbc.gridwidth = labelText.isEmpty() ? 2 : 2;
gbc.fill = GridBagConstraints.HORIZONTAL;
panel.add(field, gbc);

gbc.fill = GridBagConstraints.NONE;
}

private void setupEvents() {
    saveButton.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            saveMember();
        }
    });
}

cancelButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        dispose();
    }
});

```

```

    }

});

}

private void populateFields() {

    if (member != null) {

        namaField.setText(member.getNama());

        alamatArea.setText(member.getAlamat());

        noTeleponField.setText(member.getNoTelepon());

        emailField.setText(member.getEmail());

        statusComboBox.setSelectedItem(member.getStatusAktif());

    }

}

private void saveMember() {

    if (namaField.getText().trim().isEmpty()) {

        showError("Nama lengkap tidak boleh kosong!");

        namaField.requestFocus();

        return;

    }

    if (alamatArea.getText().trim().isEmpty()) {

```

```

showError("Alamat tidak boleh kosong");

alamatArea.requestFocus();

return;

}

if (noTeleponField.getText().trim().isEmpty()) {

showError("No. telepon tidak boleh kosong");

noTeleponField.requestFocus();

return;

}

if (emailField.getText().trim().isEmpty()) {

showError("Email tidak boleh kosong");

emailField.requestFocus();

return;

}

// Validasi format email sederhana

String email = emailField.getText().trim();

if (!email.contains("@") || !email.contains(".")) {

showError("Format email tidak valid");

emailField.requestFocus();

```

```

        return;

    }

    if (member == null) {

        member = new Member();

        // Set tanggal daftar untuk member baru

        member.setTanggalDaftar(Date.valueOf(LocalDate.now()));

    }

    member.setNama(namaField.getText().trim());

    member.setAlamat(alamatArea.getText().trim());

    member.setNoTelepon(noTeleponField.getText().trim());

    member.setEmail(emailField.getText().trim());

    member.setStatusAktif((String) statusComboBox.getSelectedItem());

}

confirmed = true;

dispose();

}

private void showError(String message) {

    JOptionPane.showMessageDialog(this,

        "Warning! " + message,

```

```

        "Input Error",
        JOptionPane.WARNING_MESSAGE);
    }

    public boolean isConfirmed() {
        return confirmed;
    }

    public Member getMember() {
        return member;
    }

}

```

11. Member Management Panel

```

package com.Library.Admin;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.DefaultTableCellRenderer;
import java.awt.*;
import java.awt.event.ActionEvent;

```



```

public MemberManagementPanel() {

    memberDAO = new MemberDAO();

    initComponents();

    setupLayout();

    setupEvents();

    loadMembers();

}

private void initComponents() {

    setBackground(Color.WHITE);

    // Table

    String[] columns = {"ID", "Nama", "Alamat", "No_Telepon", "Tanggal_Daftar", "Email",
    "Status_Aktif"};

    tableModel = new DefaultTableModel(columns, 0) {

        @Override

        public boolean isCellEditable(int row, int column) {

            return false; // Tabel tidak bisa diedit langsung

        }

    };

    memberTable = new JTable(tableModel);

    memberTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

    memberTable.setRowHeight(30);
}

```

```

memberTable.setFont(new Font("Arial", Font.PLAIN, 12));

memberTable.getTableHeader().setFont(new Font("Arial", Font.BOLD, 13));

memberTable.getTableHeader().setBackground(PRIMARY_GREEN);

memberTable.getTableHeader().setForeground(Color.WHITE);

memberTable.setGridColor(LIGHT_GREEN);

memberTable.setSelectionBackground(LIGHT_GREEN);

memberTable.setSelectionForeground(DARK_GREEN);

// Mengatur lebar kolom

memberTable.getColumnModel().getColumn(0).setPreferredWidth(50); // ID

memberTable.getColumnModel().getColumn(1).setPreferredWidth(150); // Nama

memberTable.getColumnModel().getColumn(2).setPreferredWidth(200); // Alamat

memberTable.getColumnModel().getColumn(3).setPreferredWidth(120); // No_Telepon

memberTable.getColumnModel().getColumn(4).setPreferredWidth(100); // Tanggal_Daftar

memberTable.getColumnModel().getColumn(5).setPreferredWidth(180); // Email

memberTable.getColumnModel().getColumn(6).setPreferredWidth(80); // Status_Aktif

// Custom renderer untuk kolom Status Aktif

memberTable.getColumnModel().getColumn(6).setCellRenderer(new
DefaultTableCellRenderer() {

    @Override

    public Component getTableCellRendererComponent(JTable table, Object value,

```

```

boolean isSelected, boolean hasFocus, int row, int column) {

    Component c = super.getTableCellRendererComponent(table, value, isSelected, hasFocus,
row, column);

    if (!isSelected) {

        if ("1".equalsIgnoreCase(value.toString())) {

            c.setBackground(new Color(220, 255, 220));

            c.setForeground(DARK_GREEN);

            setText("Aktif");

        } else if ("0".equalsIgnoreCase(value.toString())) {

            c.setBackground(new Color(255, 240, 240));

            c.setForeground(new Color(139, 0, 0));

            setText("Tidak Aktif");

        } else {

            c.setBackground(Color.WHITE);

            c.setForeground(Color.BLACK);

        }

    } else {

        if ("1".equalsIgnoreCase(value.toString())) {

            setText("Aktif");

        } else if ("0".equalsIgnoreCase(value.toString())) {

            setText("Tidak Aktif");

        }

    }

}

```

```
    }

    setHorizontalAlignment(SwingConstants.CENTER);

    return c;
}

});

// Search field

searchField = new JTextField(20);

searchField.setFont(new Font("Arial", Font.PLAIN, 14));

// Status label

statusLabel = new JLabel("Total: 0 anggota");

statusLabel.setFont(new Font("Arial", Font.BOLD, 12));

statusLabel.setForeground(DARK_GREEN);

// Buttons

addButton = new JButton("Tambah Anggota");

editButton = new JButton("Edit Anggota");

deleteButton = new JButton("Hapus Anggota");

refreshButton = new JButton("Refresh");

searchButton = new JButton("Cari");

printCardButton = new JButton("Cetak Kartu");
```

```

// Style buttons

styleButton(addButton, ACCENT_GREEN);

styleButton(editButton, new Color(255, 165, 0));

styleButton(deleteButton, new Color(139, 0, 0));

styleButton(refreshButton, PRIMARY_GREEN);

styleButton(searchButton, DARK_GREEN);

styleButton(printCardButton, new Color(138, 43, 226)); // Purple untuk print

}

private void styleButton(JButton button, Color color) {

button.setBackground(color);

button.setForeground(Color.WHITE);

button.setFont(new Font("Arial", Font.BOLD, 12));

button.setPreferredSize(new Dimension(140, 35));

button.setFocusPainted(false);

button.setCursor(new Cursor(Cursor.HAND_CURSOR));

button.setBorder(BorderFactory.createRaisedBevelBorder());

}

private void setupLayout() {

setLayout(new BorderLayout());

```

```

// Header Panel

JPanel headerPanel = new JPanel(new BorderLayout());

headerPanel.setBackground(DARK_GREEN);

headerPanel.setPreferredSize(new Dimension(0, 60));

JLabel titleLabel = new JLabel("DAFTAR ANGGOTA PERPUSTAKAAN");

titleLabel.setFont(new Font("Arial", Font.BOLD, 20));

titleLabel.setForeground(Color.WHITE);

titleLabel.setHorizontalAlignment(SwingConstants.CENTER);

headerPanel.add(titleLabel, BorderLayout.CENTER);

add(headerPanel, BorderLayout.NORTH);

// Top Panel (Search + Status)

JPanel topPanel = new JPanel(new BorderLayout());

topPanel.setBackground(BG_GREEN);

topPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

// Search Panel

JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));

searchPanel.setBackground(BG_GREEN);

```

```
JLabel searchLabel = new JLabel("Cari Anggota:");
searchLabel.setFont(new Font("Arial", Font.BOLD, 14));
searchLabel.setForeground(DARK_GREEN);
searchPanel.add(searchLabel);
searchPanel.add(Box.createHorizontalStrut(10));
searchPanel.add(searchField);
searchPanel.add(Box.createHorizontalStrut(10));
searchPanel.add(searchButton);

// Status Panel
JPanel statusPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
statusPanel.setBackground(BG_GREEN);
statusPanel.add(statusLabel);

topPanel.add(searchPanel, BorderLayout.WEST);
topPanel.add(statusPanel, BorderLayout.EAST);
add(topPanel, BorderLayout.NORTH);

// Table Panel
```

```

JSscrollPane scrollPane = new JSscrollPane(memberTable);

scrollPane.setBorder(BorderFactory.createTitledBorder(
    BorderFactory.createLineBorder(PRIMARY_GREEN, 2),
    "Daftar Anggota", 0, 0,
    new Font("Arial", Font.BOLD, 14), PRIMARY_GREEN));
scrollPane.getViewport().setBackground(Color.WHITE);

add(scrollPane, BorderLayout.CENTER);

// Button Panel

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 15, 15));

buttonPanel.setBackground(BG_GREEN);
buttonPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

buttonPanel.add(addButton);
buttonPanel.add(editButton);
buttonPanel.add(deleteButton);
buttonPanel.add(printCardButton);
buttonPanel.add(refreshButton);

add(buttonPanel, BorderLayout.SOUTH);

}

```

```
private void setupEvents() {  
    addButton.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            showAddMemberDialog();  
        }  
    });  
  
    editButton.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            editSelectedMember();  
        }  
    });  
  
    deleteButton.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            deleteSelectedMember();  
        }  
    });
```

```
refreshButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        loadMembers();  
    }  
});  
  
searchButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        searchMembers();  
    }  
});  
  
printCardButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        printMemberCard();  
    }  
});
```

```

// Event untuk Enter pada search field

searchField.addActionListener(new ActionListener() {

    @Override

    public void actionPerformed(ActionEvent e) {

        searchMembers();

    }

});

}

private void showAddMemberDialog() {

    MemberDialog dialog = new MemberDialog((Frame) SwingUtilities.getWindowAncestor(this),

        "Tambah Anggota Baru", null);

    dialog.setVisible(true);

}

if (dialog.isConfirmed()) {

    Member newMember = dialog.getMember();

    if (memberDAO.addMember(newMember)) {

        JOptionPane.showMessageDialog(this,

            "Anggota berhasil ditambahkan!",

            "Sukses",

            JOptionPane.INFORMATION_MESSAGE);

        loadMembers();
    }
}

```

```

} else {

    JOptionPane.showMessageDialog(this,
        "Gagal menambahkan anggota!",
        "Error",
        JOptionPane.ERROR_MESSAGE);

}

}

}

private void editSelectedMember() {

    int selectedRow = memberTable.getSelectedRow();

    if (selectedRow == -1) {

        JOptionPane.showMessageDialog(this,
            "Pilih anggota yang akan diedit!",
            "Peringatan",
            JOptionPane.WARNING_MESSAGE);

        return;
    }

    int memberId = (Integer) tableModel.getValueAt(selectedRow, 0);

    Member selectedMember = findMemberById(memberId);
}

```

```

if (selectedMember != null) {

    MemberDialog dialog = new MemberDialog((Frame)
SwingUtilities.getWindowAncestor(this),

    "Edit Anggota", selectedMember);

dialog.setVisible(true);

}

if (dialog.isConfirmed()) {

    Member updatedMember = dialog.getMember();

    if (memberDAO.updateMember(updatedMember)) {

        JOptionPane.showMessageDialog(this,
        "Anggota berhasil diperbarui!",
        "Sukses",
        JOptionPane.INFORMATION_MESSAGE);

        loadMembers();

    } else {

        JOptionPane.showMessageDialog(this,
        "Gagal memperbarui anggota!",
        "Error",
        JOptionPane.ERROR_MESSAGE);

    }

}

}

```

```

private void deleteSelectedMember() {

    int selectedRow = memberTable.getSelectedRow();

    if (selectedRow == -1) {

        JOptionPane.showMessageDialog(this,
            "Pilih anggota yang akan dihapus!",
            "Peringatan",
            JOptionPane.WARNING_MESSAGE);

        return;
    }

    int memberId = (Integer) tableModel.getValueAt(selectedRow, 0);

    String memberName = (String) tableModel.getValueAt(selectedRow, 1);

    int confirm = JOptionPane.showConfirmDialog(this,
        "Apakah Anda yakin ingin menghapus anggota:\n" + memberName + " (ID: " + memberId +
    ")?",

        "Konfirmasi Hapus",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE);

    if (confirm == JOptionPane.YES_OPTION) {

        if (memberDAO.deleteMember(memberId)) {

```

```

JOptionPane.showMessageDialog(this,
    "Anggota berhasil dihapus!",
    "Sukses",
    JOptionPane.INFORMATION_MESSAGE);

loadMembers();

} else {

JOptionPane.showMessageDialog(this,
    "Gagal menghapus anggota!",
    "Error",
    JOptionPane.ERROR_MESSAGE);

}

}

}

private void searchMembers() {

String keyword = searchField.getText().trim();

if (keyword.isEmpty()) {

loadMembers();

return;

}

List<Member> searchResults = memberDAO.searchMembers(keyword);

```

```

populateTable(searchResults);

updateStatusLabel(searchResults.size());

}

private void printMemberCard() {

    int selectedRow = memberTable.getSelectedRow();

    if (selectedRow == -1) {

        JOptionPane.showMessageDialog(this,
            "Pilih anggota untuk mencetak kartu!",
            "Peringatan",
            JOptionPane.WARNING_MESSAGE);

        return;
    }

    int memberId = (Integer) tableModel.getValueAt(selectedRow, 0);

    Member selectedMember = findMemberById(memberId);

    if (selectedMember != null) {

        printCard(selectedMember);
    }
}

```

```

private void printCard(Member member) {

    PrinterJob job = PrinterJob.getPrinterJob();

    job.setPrintable(new MemberCardPrintable(member));

}

if(job.printDialog()) {

    try {

        job.print();

        JOptionPane.showMessageDialog(this,
            "Kartu anggota berhasil dicetak!",
            "Sukses",
            JOptionPane.INFORMATION_MESSAGE);

    } catch (PrinterException e) {

        JOptionPane.showMessageDialog(this,
            "Gagal mencetak kartu: " + e.getMessage(),
            "Error",
            JOptionPane.ERROR_MESSAGE);

    }

}

}

private void loadMembers() {

    List<Member> members = memberDAO.getAllMembers();
}

```

```

populateTable(members);

updateStatusLabel(members.size());

}

private void populateTable(List<Member> members) {

    tableModel.setRowCount(0);

    for (Member member : members) {

        Object[] row = {

            member.getIdAnggota(),
            member.getNama(),
            member.getAlamat(),
            member.getNoTelepon(),
            member.getTanggalDaftar(),
            member.getEmail(),
            member.getStatusAktif()
        };

        tableModel.addRow(row);
    }
}

private void updateStatusLabel(int count) {

    int activeMembers = memberDAO.getActiveMembers();
}

```

```

int inactiveMembers = memberDAO.getInactiveMembers();

statusLabel.setText(String.format("Total: %d anggota | Aktif: %d | Tidak Aktif: %d",
                                count, activeMembers, inactiveMembers));

}

private Member findMemberById(int id) {
    List<Member> members = memberDAO.getAllMembers();

    for (Member member : members) {
        if (member.getIdAnggota() == id) {
            return member;
        }
    }

    return null;
}

// Inner class untuk mencetak kartu anggota

private class MemberCardPrintable implements Printable {
    private Member member;

    public MemberCardPrintable(Member member) {
        this.member = member;
    }
}

```

```

@Override
public int print(Graphics graphics, PageFormat pageFormat, int pageIndex)
throws PrinterException {
if (pageIndex > 0) {
return NO_SUCH_PAGE;
}

Graphics2D g2d = (Graphics2D) graphics;
g2d.translate(pageFormat.getImageableX(), pageFormat.getImageableY());

// Enable antialiasing untuk hasil yang lebih halus
g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

g2d.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
RenderingHints.VALUE_TEXT_ANTIALIAS_ON);

int cardWidth = 400;
int cardHeight = 280;
int margin = 50;

// Draw card border dengan rounded rectangle
g2d.setColor(new Color(102, 165, 130));

```

```
g2d.setStroke(new BasicStroke(3));

g2d.drawRoundRect(margin, margin, cardWidth, cardHeight, 15, 15);

// Header background dengan gradient

GradientPaint headerGradient = new GradientPaint(
    margin, margin, new Color(102, 165, 130),
    margin, margin + 60, new Color(139, 195, 158)
);

g2d.setPaint(headerGradient);

g2d.fillRoundRect(margin + 2, margin + 2, cardWidth - 4, 58, 12, 12);

// Informasi perpustakaan (header)

g2d.setColor(Color.WHITE);

g2d.setFont(new Font("Arial", Font.BOLD, 18));

g2d.drawString("PERPUSTAKAAN MARI MACA", margin + 20, margin + 25);

g2d.setFont(new Font("Arial", Font.PLAIN, 11));

g2d.drawString("Jl. Galuh Mas No. 123, Karawang Barat 10430", margin + 20, margin + 42);

g2d.drawString("Telp: (021) 1234-5678 | Email: marimaca@perpusdigital.id", margin + 20,
margin + 55);

// Title kartu anggota
```

```

g2d.setColor(new Color(102, 165, 130)); // Soft Green

g2d.setFont(new Font("Arial", Font.BOLD, 16));

g2d.drawString("KARTU ANGGOTA", margin + 20, margin + 85);

// Garis pemisah

g2d.setStroke(new BasicStroke(1));

g2d.drawLine(margin + 20, margin + 95, margin + cardWidth - 20, margin + 95);

// Data anggota

g2d.setColor(Color.BLACK);

int startY = margin + 115;

int lineHeight = 16;

// ID Anggota dengan highlight

g2d.setFont(new Font("Arial", Font.BOLD, 12));

g2d.drawString("ID ANGGOTA", margin + 20, startY);

g2d.setFont(new Font("Arial", Font.BOLD, 14));

g2d.setColor(new Color(220, 20, 60)); // Crimson untuk ID

g2d.drawString(String.valueOf(member.getIdAnggota()), margin + 120, startY);

// Reset warna

g2d.setColor(Color.BLACK);

```

```
g2d.setFont(new Font("Arial", Font.PLAIN, 11));  
  
startY += lineHeight + 3;  
  
g2d.drawString("Nama", margin + 20, startY);  
  
g2d.setFont(new Font("Arial", Font.BOLD, 11));  
  
g2d.drawString(": " + member.getNama(), margin + 120, startY);  
  
startY += lineHeight;  
  
g2d.setFont(new Font("Arial", Font.PLAIN, 11));  
  
g2d.drawString("Alamat", margin + 20, startY);  
  
g2d.drawString(": " + member.getAlamat(), margin + 120, startY);  
  
startY += lineHeight;  
  
g2d.drawString("No. Telepon", margin + 20, startY);  
  
g2d.drawString(": " + member.getNoTelepon(), margin + 120, startY);  
  
startY += lineHeight;  
  
g2d.drawString("Email", margin + 20, startY);  
  
g2d.drawString(": " + member.getEmail(), margin + 120, startY);  
  
startY += lineHeight;  
  
g2d.drawString("Tgl. Daftar", margin + 20, startY);
```

```

g2d.drawString(": " + member.getTanggalDaftar(), margin + 120, startY);

startY += lineHeight + 3;

String status = "1".equals(member.getStatusAktif()) ? "AKTIF" : "TIDAK AKTIF";

g2d.drawString("Status", margin + 20, startY);

// Status dengan warna berbeda

if ("AKTIF".equals(status)) {

    g2d.setColor(new Color(76, 153, 102)); // Darker Soft Green untuk AKTIF

} else {

    g2d.setColor(new Color(220, 20, 60)); // Crimson tetap untuk TIDAK AKTIF

}

g2d.setFont(new Font("Arial", Font.BOLD, 11));

g2d.drawString(": " + status, margin + 120, startY);

// Footer dengan garis dan informasi tambahan

g2d.setColor(new Color(102, 165, 130));

g2d.setStroke(new BasicStroke(2));

int footerLineY = margin + cardHeight - 55; // Posisi garis footer

g2d.drawLine(margin + 20, footerLineY, margin + cardWidth - 20, footerLineY);

```

```

// Teks footer

g2d.setFont(new Font("Arial", Font.ITALIC, 9));

g2d.setColor(new Color(100, 100, 100));

g2d.drawString("Kartu ini adalah bukti keanggotaan resmi dan berlaku selama status aktif",

margin + 20, footerLineY + 18);

g2d.drawString("Harap dibawa setiap kali mengunjungi perpustakaan",

margin + 20, footerLineY + 30);

// Nomor seri kartu

g2d.setFont(new Font("Arial", Font.PLAIN, 8));

g2d.setColor(Color.LIGHT_GRAY);

String serialNumber = "PD-" + member.getIdAnggota() + "-" +

member.getTanggalDaftar().toString().replace("-", "").substring(2);

g2d.drawString("No. Seri: " + serialNumber, margin + cardWidth - 120, margin + cardHeight -

8);

return PAGE_EXISTS;

}

}

}

```

12. Peminjaman

```
package com.Library.Admin;
```

```
import java.sql.Date;

public class Peminjaman {

    private int idPeminjaman;

    private int idBuku;

    private int idAnggota;

    private Date tanggalPinjam;

    private Date tanggalJatuhTempo;

    private Date tanggalPengembalian;

    private String statusPeminjaman;

    private String judulBuku;

    private String namaAnggota;

    public Peminjaman() {

    }

    public Peminjaman(int idPeminjaman, int idBuku, int idAnggota, Date tanggalPinjam, Date
        tanggalJatuhTempo, Date tanggalPengembalian, String statusPeminjaman, String judulBuku, String
        namaAnggota) {

        this.idPeminjaman = idPeminjaman;

        this.idBuku = idBuku;
```

```
this.idAnggota = idAnggota;  
  
this.tanggalPinjam = tanggalPinjam;  
  
this.tanggalJatuhTempo = tanggalJatuhTempo;  
  
this.tanggalPengembalian = tanggalPengembalian;  
  
this.statusPeminjaman = statusPeminjaman;  
  
this.judulBuku = judulBuku;  
  
this.namaAnggota = namaAnggota;  
  
/ Getters and Setters  
  
public int getIdPeminjaman() {  
    return idPeminjaman;  
}  
  
public void setIdPeminjaman(int idPeminjaman) {  
    this.idPeminjaman = idPeminjaman;  
}  
  
public int getIdBuku() {  
    return idBuku;
```

```
public void setIdBuku(int idBuku) {  
    this.idBuku = idBuku;  
}  
  
public int getIdAnggota() {  
    return idAnggota;  
}  
  
public void setIdAnggota(int idAnggota) {  
    this.idAnggota = idAnggota;  
}  
  
public Date getTanggalPinjam() {  
    return tanggalPinjam;  
}  
  
public void setTanggalPinjam(Date tanggalPinjam) {  
    this.tanggalPinjam = tanggalPinjam;  
}  
  
public Date getTanggalJatuhTempo() {  
    return tanggalJatuhTempo;
```

```
}

public void setTanggalJatuhTempo(Date tanggalJatuhTempo) {
    this.tanggalJatuhTempo = tanggalJatuhTempo;
}

public Date getTanggalPengembalian() {
    return tanggalPengembalian;
}

public void setTanggalPengembalian(Date tanggalPengembalian) {
    this.tanggalPengembalian = tanggalPengembalian;
}

public String getStatusPeminjaman() {
    return statusPeminjaman;
}

public void setStatusPeminjaman(String statusPeminjaman) {
    this.statusPeminjaman = statusPeminjaman;
}
```

```

public String getJudulBuku() {

    return judulBuku;

}

public void setJudulBuku(String judulBuku) {

    this.judulBuku = judulBuku;

}

public String getNamaAnggota() {

    return namaAnggota;

}

public void setNamaAnggota(String namaAnggota) {

    this.namaAnggota = namaAnggota;

}

@Override

public String toString() {

    return "Peminjaman{" +
        "idPeminjaman=" + idPeminjaman +
        ", idBuku=" + idBuku +
        ", idAnggota=" + idAnggota +
}

```

```

", tanggalPinjam=" + tanggalPinjam +
", tanggalJatuhTempo=" + tanggalJatuhTempo +
", tanggalPengembalian=" + tanggalPengembalian +
", statusPeminjaman="" + statusPeminjaman + \'\" +
", judulBuku="" + judulBuku + \'\" +
", namaAnggota="" + namaAnggota + \'\" +
'}';
}

```

13. Peminjaman DAO (Data Access Object)

```

package com.Library.Admin;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PeminjamanDAO {
    private Connection connection;
    public PeminjamanDAO() {

```

```

this.connection = DBConnection.getConnection();

}

public boolean addPeminjaman(Peminjaman peminjaman) {

    String sqlPeminjaman = "INSERT INTO peminjaman (id_buku, id_anggota, tanggal_pinjam,
    tanggal_jatuh_tempo, status_peminjaman) VALUES (?, ?, ?, ?, ?)";

    boolean success = false;

    try {

        // Nonaktifkan auto-commit untuk transaksi

        connection.setAutoCommit(false);

    }

    PreparedStatement pstmtPeminjaman = connection.prepareStatement(sqlPeminjaman,
Statement.RETURN_GENERATED_KEYS);

    pstmtPeminjaman.setInt(1, peminjaman.getIdBuku());

    pstmtPeminjaman.setInt(2, peminjaman.getIdAnggota());

    pstmtPeminjaman.setDate(3, peminjaman.getTanggalPinjam());

    pstmtPeminjaman.setDate(4, peminjaman.getTanggalJatuhTempo());

    pstmtPeminjaman.setString(5, peminjaman.getStatusPeminjaman());

    int resultPeminjaman = pstmtPeminjaman.executeUpdate();

    if (resultPeminjaman > 0) {

        String sqlUpdateBuku = "UPDATE buku SET status = 'Dipinjam' WHERE id_buku = ?";

        PreparedStatement pstmtUpdateBuku = connection.prepareStatement(sqlUpdateBuku);

```

```

        pstmtUpdateBuku.setInt(1, peminjaman.getIdBuku());

        int resultUpdateBuku = pstmtUpdateBuku.executeUpdate();

        if (resultUpdateBuku > 0) {

            connection.commit(); // Commit transaksi jika semua berhasil

            success = true;

        } else {

            connection.rollback(); // Rollback jika update status buku gagal

            System.err.println("Gagal mengupdate status buku. Transaksi dirollback.");

        }

        pstmtUpdateBuku.close();

    } else {

        connection.rollback(); // Rollback jika insert peminjaman gagal

        System.err.println("Gagal menambahkan peminjaman. Transaksi dirollback.");

    }

    pstmtPeminjaman.close();

} catch (SQLException e) {

    try {

        connection.rollback(); // Rollback jika terjadi exception

        System.err.println("SQLException terjadi. Transaksi dirollback.");

    } catch (SQLException ex) {

        ex.printStackTrace();

```

```

    }

    e.printStackTrace();

} finally {

try {

connection.setAutoCommit(true); // Kembalikan auto-commit ke true

} catch (SQLException ex) {

ex.printStackTrace();

}

}

return success;
}

public List<Peminjaman> getAllPeminjaman() {

List<Peminjaman> daftarPeminjaman = new ArrayList<>();

String sql = "SELECT p.id_peminjaman, p.id_buku, b.judul_buku, p.id_anggota, a.nama AS
nama_anggota, " +

"p.tanggal_pinjam, p.tanggal_jatuh_tempo, p.tanggal_pengembalian,
p.status_peminjaman " +

"FROM peminjaman p " +

"JOIN buku b ON p.id_buku = b.id_buku " +

"JOIN anggota a ON p.id_anggota = a.id_anggota " +

"ORDER BY p.tanggal_pinjam DESC, p.id_peminjaman DESC";

```

```

try (Statement stmt = connection.createStatement());

    ResultSet rs = stmt.executeQuery(sql)) {

}

while (rs.next()) {

    Peminjaman peminjaman = new Peminjaman();

    peminjaman.setIdPeminjaman(rs.getInt("id_peminjaman"));

    peminjaman.setIdBuku(rs.getInt("id_buku"));

    peminjaman.setJudulBuku(rs.getString("judul_buku"));

    peminjaman.setIdAnggota(rs.getInt("id_anggota"));

    peminjaman.setNamaAnggota(rs.getString("nama_anggota"));

    peminjaman.setTanggalPinjam(rs.getDate("tanggal_pinjam"));

    peminjaman.setTanggalJatuhTempo(rs.getDate("tanggal_jatuh_tempo"));

    peminjaman.setTanggalPengembalian(rs.getDate("tanggal_pengembalian"));

    peminjaman.setStatusPeminjaman(rs.getString("status_peminjaman"));

    daftarPeminjaman.add(peminjaman);

}

} catch (SQLException e) {

    e.printStackTrace();

}

return daftarPeminjaman;
}

```

```

public boolean updatePeminjaman(Peminjaman peminjaman) {

    String sqlUpdatePeminjaman = "UPDATE peminjaman SET id_buku=?, id_anggota=?,
tanggal_pinjam=?, " +
                                "tanggal_jatuh_tempo=?, tanggal_pengembalian=?, status_peminjaman=? " +
                                "WHERE id_peminjaman=?";

    boolean success = false;

    try {
        connection.setAutoCommit(false);

        PreparedStatement pstmt = connection.prepareStatement(sqlUpdatePeminjaman);

        pstmt.setInt(1, peminjaman.getIdBuku());
        pstmt.setInt(2, peminjaman.getIdAnggota());
        pstmt.setDate(3, peminjaman.getTanggalPinjam());
        pstmt.setDate(4, peminjaman.getTanggalJatuhTempo());
        pstmt.setDate(5, peminjaman.getTanggalPengembalian());
        pstmt.setString(6, peminjaman.getStatusPeminjaman());
        pstmt.setInt(7, peminjaman.getIdPeminjaman());

        int resultUpdate = pstmt.executeUpdate();

        if (resultUpdate > 0) {
            if ("Dikembalikan".equalsIgnoreCase(peminjaman.getStatusPeminjaman())) {
                String sqlUpdateBuku = "UPDATE buku SET status = 'Tersedia' WHERE id_buku = ?";
                PreparedStatement pstmtUpdateBuku = connection.prepareStatement(sqlUpdateBuku);
            }
        }
    }
}

```

```

        pstmtUpdateBuku.setInt(1, peminjaman.getIdBuku());

        int resultUpdateBuku = pstmtUpdateBuku.executeUpdate();

        if (resultUpdateBuku > 0) {

            connection.commit();

            success = true;

        } else {

            connection.rollback();

            System.err.println("Gagal mengupdate status buku saat pengembalian. Transaksi dirollback.");
        }

        pstmtUpdateBuku.close();

    } else if ("Dipinjam".equalsIgnoreCase(peminjaman.getStatusPeminjaman())) {

        // Jika status diubah (misal dari Terlambat) kembali ke Dipinjam atau memang Dipinjam saat update lain

        String sqlUpdateBukuStatusToDipinjam = "UPDATE buku SET status = 'Dipinjam' WHERE id_buku = ?";

        PreparedStatement pstmtUpdateBukuStatus = connection.prepareStatement(sqlUpdateBukuStatusToDipinjam);

        pstmtUpdateBukuStatus.setInt(1, peminjaman.getIdBuku());

        int resultUpdateBukuStatus = pstmtUpdateBukuStatus.executeUpdate();

        if (resultUpdateBukuStatus > 0) {

            connection.commit();

            success = true;

        } else {
    }

```

```

        connection.rollback();

        System.err.println("Gagal mengupdate status buku menjadi 'Dipinjam'. Transaksi
dirollback.");
    }

    pstmtUpdateBukuStatus.close();

}

else {

    connection.commit();

    success = true;

}

} else {

    connection.rollback();

    System.err.println("Gagal mengupdate peminjaman. Transaksi dirollback.");

}

pstmt.close();

} catch (SQLException e) {

    try { connection.rollback(); } catch (SQLException ex) { ex.printStackTrace(); }

    e.printStackTrace();

} finally {

    try { connection.setAutoCommit(true); } catch (SQLException ex) { ex.printStackTrace(); }

}

return success;
}

```

```

public boolean deletePeminjaman(int idPeminjaman) {

    String sql = "DELETE FROM peminjaman WHERE id_peminjaman = ?";

    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

        pstmt.setInt(1, idPeminjaman);

        int result = pstmt.executeUpdate();

        return result > 0;

    } catch (SQLException e) {

        System.err.println("SQLException saat mencoba menghapus peminjaman dengan ID: " +
idPeminjaman);

        e.printStackTrace();

        return false; // Mengembalikan false jika terjadi kesalahan SQL

    }

}

}

public List<Peminjaman> searchPeminjaman(String keyword) {

    List<Peminjaman> daftarPeminjaman = new ArrayList<>();

    String sql = "SELECT p.id_peminjaman, p.id_buku, b.judul_buku, p.id_anggota, a.nama AS
nama_anggota, " +

            "p.tanggal_pinjam, p.tanggal_jatuh_tempo, p.tanggal_pengembalian,
p.status_peminjaman " +

            "FROM peminjaman p " +

            "JOIN buku b ON p.id_buku = b.id_buku " +

            "JOIN anggota a ON p.id_anggota = a.id_anggota " +

```

```

"WHERE b.judul_buku LIKE ? OR a.nama LIKE ? OR p.status_peminjaman LIKE ? " +
"ORDER BY p.tanggal_pinjam DESC, p.id_peminjaman DESC";

try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

    String searchPattern = "%" + keyword + "%";
    pstmt.setString(1, searchPattern);
    pstmt.setString(2, searchPattern);
    pstmt.setString(3, searchPattern);

    ResultSet rs = pstmt.executeQuery();

    while (rs.next()) {

        Peminjaman peminjaman = new Peminjaman();
        peminjaman.setIdPeminjaman(rs.getInt("id_peminjaman"));
        peminjaman.setIdBuku(rs.getInt("id_buku"));
        peminjaman.setJudulBuku(rs.getString("judul_buku"));
        peminjaman.setIdAnggota(rs.getInt("id_anggota"));
        peminjaman.setNamaAnggota(rs.getString("nama_anggota"));
        peminjaman.setTanggalPinjam(rs.getDate("tanggal_pinjam"));
        peminjaman.setTanggalJatuhTempo(rs.getDate("tanggal_jatuh_tempo"));
        peminjaman.setTanggalPengembalian(rs.getDate("tanggal_pengembalian"));
        peminjaman.setStatusPeminjaman(rs.getString("status_peminjaman"));

        daftarPeminjaman.add(peminjaman);
    }
}

```

```

    }

    rs.close();

} catch (SQLException e) {

    e.printStackTrace();

}

return daftarPeminjaman;
}

public int getTotalPeminjamanAktif() {

    String sql = "SELECT COUNT(*) as total FROM peminjaman WHERE status_peminjaman = 'Dipinjam'";

    try (Statement stmt = connection.createStatement());

        ResultSet rs = stmt.executeQuery(sql)) {

            if (rs.next()) {

                return rs.getInt("total");

            }

        }

    } catch (SQLException e) {

        e.printStackTrace();

    }

    return 0;
}

// Metode baru untuk mengambil buku yang tersedia

```

```

public List<PeminjamanDialog.BukuItem> getAvailableBooks() {

    List<PeminjamanDialog.BukuItem> books = new ArrayList<>();

    String sql = "SELECT id_buku, judul_buku FROM buku WHERE status = 'Tersedia' ORDER BY
id_buku";

    try (Statement stmt = connection.createStatement()) {

        ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {

                // Format tampilan: "ID - Judul Buku"

                String display = rs.getInt("id_buku") + " - " + rs.getString("judul_buku");

                books.add(new PeminjamanDialog.BukuItem(rs.getInt("id_buku"), display));

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return books;
    }

    // Metode untuk mengambil anggota yang aktif

    public List<PeminjamanDialog.AnggotaItem> getActiveMembers() {

        List<PeminjamanDialog.AnggotaItem> members = new ArrayList<>();

        String sql = "SELECT id_anggota, nama FROM anggota WHERE status_aktif = '1' ORDER BY
id_anggota";

        try (Statement stmt = connection.createStatement());

```

```

ResultSet rs = stmt.executeQuery(sql)) {

    while (rs.next()) {

        // Format tampilan: "ID - Nama Anggota"

        String display = rs.getInt("id_anggota") + " - " + rs.getString("nama");

        members.add(new PeminjamanDialog.AnggotaItem(rs.getInt("id_anggota"), display));

    }

} catch (SQLException e) {

    e.printStackTrace();

}

return members;

}
}

```

14. Peminjaman Dialog

```

package com.Library.Admin;

import javax.swing.*;
import java.awt.*;
import java.sql.Date;
import java.text.ParseException;
import java.text.SimpleDateFormat;

```

```
import java.util.List;  
  
import java.util.Objects;  
  
public class PeminjamanDialog extends JDialog {  
  
    private JComboBox<BukuItem> bukuComboBox;  
  
    private JComboBox<AnggotaItem> anggotaComboBox;  
  
    private JTextField tanggalPinjamField;  
  
    private JTextField tanggalJatuhTempoField;  
  
    private JTextField tanggalPengembalianField;  
  
    private JComboBox<String> statusComboBox;  
  
    private JButton saveButton;  
  
    private JButton cancelButton;  
  
    private boolean confirmed = false;  
  
    private Peminjaman peminjaman;  
  
    private PeminjamanDAO peminjamanDAO;  
  
    private SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");  
  
    private final Color PRIMARY_GREEN = new Color(34, 139, 34);  
  
    private final Color DARK_GREEN = new Color(0, 100, 0);  
  
    private final Color BG_GREEN = new Color(240, 255, 240);
```

```
// Inner class untuk item di JComboBox Buku

public static class BukuItem {
    private int id;
    private String displayValue;

    public BukuItem(int id, String displayValue) {
        this.id = id;
        this.displayValue = displayValue;
    }

    public int getId() {
        return id;
    }

    @Override
    public String toString() {
        return displayValue; // Ini yang akan ditampilkan di JComboBox
    }

    @Override
    public boolean equals(Object o) {
```

```

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

BukuItem bukuItem = (BukuItem) o;

return id == bukuItem.id;

}

@Override

public int hashCode() {

    return Objects.hash(id);

}

}

// Inner class untuk item di JComboBox Anggota

public static class AnggotaItem {

    private int id;

    private String displayValue;

    public AnggotaItem(int id, String displayValue) {

        this.id = id;

        this.displayValue = displayValue;

    }

}

```

```
public int getId() {  
    return id;  
}  
  
@Override  
public String toString() {  
    return displayValue; // Ini yang akan ditampilkan di JComboBox  
}  
  
@Override  
public boolean equals(Object o) {  
    if (this == o) return true;  
    if (o == null || getClass() != o.getClass()) return false;  
    AnggotaItem that = (AnggotaItem) o;  
    return id == that.id;  
}  
  
@Override  
public int hashCode() {  
    return Objects.hash(id);  
}
```

```
public PeminjamanDialog(Frame parent, String title, Peminjaman peminjamanToEdit) {  
    super(parent, title, true);  
    this.peminjaman = peminjamanToEdit;  
    this.peminjamanDAO = new PeminjamanDAO();  
  
    initComponents();  
    setupLayout();  
    setupEvents();  
  
    if (peminjamanToEdit != null) { // Mode Edit  
        populateFields(peminjamanToEdit);  
        bukuComboBox.setEnabled(false); // Tidak bisa ganti buku saat edit  
        anggotaComboBox.setEnabled(false); // Tidak bisa ganti anggota saat edit  
        if ("Dikembalikan".equalsIgnoreCase(peminjamanToEdit.getStatusPeminjaman())) {  
            tanggalPinjamField.setEditable(false);  
            tanggalJatuhTempoField.setEditable(false);  
            statusComboBox.setEnabled(false); // Jika sudah kembali, status juga tidak bisa diubah lagi  
            sembarang  
        }  
    } else {  
        loadAvailableBooksToComboBox();  
    }  
}
```

```

loadActiveMembersToComboBox();

tanggalPinjamField.setText(dateFormat.format(new java.util.Date())); // Tanggal hari ini

statusComboBox.setSelectedItem("Dipinjam");

tanggalPengembalianField.setEnabled(false); // Nonaktifkan saat baru

bukuComboBox.setEnabled(true);

anggotaComboBox.setEnabled(true);

}

}

private void initComponents() {

setSize(600, 550);

setLocationRelativeTo(getParent());

setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);

}

bukuComboBox = new JComboBox<>();

anggotaComboBox = new JComboBox<>();

tanggalPinjamField = new JTextField(15);

tanggalJatuhTempoField = new JTextField(15);

tanggalPengembalianField = new JTextField(15);

String[] statusOptions = {"Dipinjam", "Dikembalikan", "Terlambat"};

statusComboBox = new JComboBox<>(statusOptions);

```

```

saveButton = new JButton("Simpan");

cancelButton = new JButton("Batal");

// Styling

styleComboBox(bukuComboBox); // Style untuk JComboBox

styleComboBox(anggotaComboBox); // Style untuk JComboBox

styleTextField(tanggalPinjamField);

styleTextField(tanggalJatuhTempoField);

styleTextField(tanggalPengembalianField);

styleComboBox(statusComboBox);

styleButton(saveButton, PRIMARY_GREEN);

styleButton(cancelButton, new Color(220, 20, 60));

}

private void loadAvailableBooksToComboBox() {

bukuComboBox.removeAllItems(); // Kosongkan dulu

bukuComboBox.addItem(new BukuItem(0, "--- Pilih Buku ---")); // Placeholder

List<BukuItem> availableBooks = peminjamanDAO.getAvailableBooks();

for (BukuItem book : availableBooks) {

bukuComboBox.addItem(book);

}

```

```

}

private void loadActiveMembersToComboBox() {
    anggotaComboBox.removeAllItems(); // Kosongkan dulu
    anggotaComboBox.addItem(new AnggotaItem(0, "--- Pilih Anggota ---")); // Placeholder
    List<AnggotaItem> activeMembers = peminjamanDAO.getActiveMembers();
    for (AnggotaItem member : activeMembers) {
        anggotaComboBox.addItem(member);
    }
}

private void styleTextField(JTextField field) {
    field.setFont(new Font("Arial", Font.PLAIN, 14));
    field.setBorder(BorderFactory.createCompoundBorder(
        BorderFactory.createLineBorder(PRIMARY_GREEN, 1),
        BorderFactory.createEmptyBorder(5, 8, 5, 8)
    ));
}

private void styleComboBox(JComboBox<?> comboBox) {
    comboBox.setFont(new Font("Arial", Font.PLAIN, 14));
    comboBox.setBackground(Color.WHITE);
}

```

```

comboBox.setBorder(BorderFactory.createLineBorder(PRIMARY_GREEN, 1));

}

private void styleButton(JButton button, Color color) {
    button.setBackground(color);
    button.setForeground(Color.WHITE);
    button.setFont(new Font("Arial", Font.BOLD, 14));
    button.setPreferredSize(new Dimension(120, 40));
    button.setFocusPainted(false);
    button.setCursor(new Cursor(Cursor.HAND_CURSOR));
    button.setBorder(BorderFactory.createRaisedBevelBorder());
}

}

private void setupLayout() {
    setLayout(new BorderLayout());
}

JPanel headerPanel = new JPanel();
headerPanel.setBackground(DARK_GREEN);
headerPanel.setPreferredSize(new Dimension(0, 60));
JLabel titleLabelDialog = new JLabel getTitle();
titleLabelDialog.setFont(new Font("Arial", Font.BOLD, 18));
titleLabelDialog.setForeground(Color.WHITE);

```

```

titleLabelDialog.setHorizontalAlignment(SwingConstants.CENTER);

headerPanel.add(titleLabelDialog);

add(headerPanel, BorderLayout.NORTH);

JPanel formPanel = new JPanel(new GridBagLayout());

formPanel.setBackground(BG_GREEN);

formPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(8, 8, 8, 8);

gbc.anchor = GridBagConstraints.WEST;

addFormField(formPanel, gbc, 0, "Buku:", bukuComboBox);

addFormField(formPanel, gbc, 1, "Anggota:", anggotaComboBox);

addFormField(formPanel, gbc, 2, "Tgl Pinjam (YYYY-MM-DD):", tanggalPinjamField);

addFormField(formPanel, gbc, 3, "Tgl Jatuh Tempo (YYYY-MM-DD):", tanggalJatuhTempoField);

addFormField(formPanel, gbc, 4, "Status Peminjaman:", statusComboBox);

addFormField(formPanel, gbc, 5, "Tgl Pengembalian (YYYY-MM-DD):", tanggalPengembalianField);

add(formPanel, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 20));

```

```

buttonPanel.setBackground(BG_GREEN);

buttonPanel.add(saveButton);

buttonPanel.add(cancelButton);

add(buttonPanel, BorderLayout.SOUTH);

}

private void addFormField(JPanel panel, GridBagConstraints gbc, int row, String labelText,
JComponent field) {

JLabel label = new JLabel(labelText);

label.setFont(new Font("Arial", Font.BOLD, 14));

label.setForeground(DARK_GREEN);

gbc.gridx = 0;

gbc.gridy = row;

gbc.gridwidth = 1;

panel.add(label, gbc);

}

gbc.gridx = 1;

gbc.gridy = row;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.HORIZONTAL;

panel.add(field, gbc);

gbc.fill = GridBagConstraints.NONE;

}

```

```

private void setupEvents() {

    saveButton.addActionListener(e -> savePeminjamanData());

    cancelButton.addActionListener(e -> dispose());
}

statusComboBox.addActionListener(e -> {

    String selectedStatus = (String) statusComboBox.getSelectedItem();

    if ("Dikembalikan".equalsIgnoreCase(selectedStatus)) {

        tanggalPengembalianField.setEnabled(true);

        if (tanggalPengembalianField.getText().trim().isEmpty()) {

            tanggalPengembalianField.setText(dateFormat.format(new java.util.Date())); // Default ke
hari ini

        }

    } else {

        tanggalPengembalianField.setEnabled(false);

        tanggalPengembalianField.setText(""); // Kosongkan jika bukan dikembalikan

    }

});

}

private void populateFields(Peminjaman p) {

// Untuk mode edit, buku dan anggota sudah ditentukan dan tidak bisa diubah

// Tampilkan buku yang sedang dipinjam

```

```

bukuComboBox.removeAllItems();

String bukuDisplay = (p.getJudulBuku() != null) ? p.getIdBuku() + " - " + p.getJudulBuku() : "ID: " + p.getIdBuku();

bukuComboBox.addItem(new BukuItem(p.getIdBuku(), bukuDisplay));

bukuComboBox.setSelectedIndex(0);

// Tampilkan anggota yang meminjam

anggotaComboBox.removeAllItems();

String anggotaDisplay = (p.getNamaAnggota() != null) ? p.getIdAnggota() + " - " + p.getNamaAnggota() : "ID: " + p.getIdAnggota();

anggotaComboBox.addItem(new AnggotaItem(p.getIdAnggota(), anggotaDisplay));

anggotaComboBox.setSelectedIndex(0);

tanggalPinjamField.setText(p.getTanggalPinjam() != null ? dateFormat.format(p.getTanggalPinjam()) : "");

tanggalJatuhTempoField.setText(p.getTanggalJatuhTempo() != null ? dateFormat.format(p.getTanggalJatuhTempo()) : "");

statusComboBox.setSelectedItem(p.getStatusPeminjaman());

tanggalPengembalianField.setText(p.getTanggalPengembalian() != null ? dateFormat.format(p.getTanggalPengembalian()) : "");

if ("Dikembalikan".equalsIgnoreCase(p.getStatusPeminjaman())) {

    tanggalPengembalianField.setEnabled(true);

} else {

```

```

tanggalPengembalianField.setEnabled(false);

}

}

private Date parseDate(String dateStr) {

    if (dateStr == null || dateStr.trim().isEmpty()) {

        return null;

    }

    try {

        java.util.Date utilDate = dateFormat.parse(dateStr.trim());

        return new Date(utilDate.getTime());

    } catch (ParseException e) {

        showError("Format tanggal salah untuk '" + dateStr + "'. Gunakan YYYY-MM-DD.");

        return null;

    }

}

private void savePeminjamanData() {

    Date tglPinjam = parseDate(tanggalPinjamField.getText());

    if (tglPinjam == null && !tanggalPinjamField.getText().trim().isEmpty()) return;

    if (tglPinjam == null) {

        showError("Tanggal Pinjam tidak boleh kosong!");

    }
}

```

```

tanggalPinjamField.requestFocus();

return;

}

Date tglJatuhTempo = parseDate(tanggalJatuhTempoField.getText());

if (tglJatuhTempo == null && !tanggalJatuhTempoField.getText().trim().isEmpty()) return;

if (tglJatuhTempo == null) {

showError("Tanggal Jatuh Tempo tidak boleh kosong!");

tanggalJatuhTempoField.requestFocus();

return;

}

String status = (String) statusComboBox.getSelectedItem();

Date tglPengembalian = null;

if ("Dikembalikan".equalsIgnoreCase(status)) {

tglPengembalian = parseDate(tanggalPengembalianField.getText());

if (tglPengembalian == null && !tanggalPengembalianField.getText().trim().isEmpty()) return;

if (tglPengembalian == null) {

showError("Tanggal Pengembalian harus diisi jika status 'Dikembalikan!'");

tanggalPengembalianField.requestFocus();

return;

}

```

```

    }

}

if (this.peminjaman == null) { // Mode Tambah Baru

    this.peminjaman = new Peminjaman();

}

BukuItem selectedBuku = (BukuItem) bukuComboBox.getSelectedItem();

if (selectedBuku == null || selectedBuku.getId() == 0) { // 0 adalah ID placeholder

    showError("Buku harus dipilih!");

    bukuComboBox.requestFocus();

    return;

}

this.peminjaman.setIdBuku(selectedBuku.getId());

}

AnggotaItem selectedAnggota = (AnggotaItem) anggotaComboBox.getSelectedItem();

if (selectedAnggota == null || selectedAnggota.getId() == 0) { // 0 adalah ID placeholder

    showError("Anggota harus dipilih!");

    anggotaComboBox.requestFocus();

    return;

}

this.peminjaman.setIdAnggota(selectedAnggota.getId());

}

}

```

```

// Untuk mode edit, ID Buku dan ID Anggota sudah ada di this.peminjaman dan tidak diubah

this.peminjaman.setTanggalPinjam(tglPinjam);
this.peminjaman.setTanggalJatuhTempo(tglJatuhTempo);
this.peminjaman.setStatusPeminjaman(status);
this.peminjaman.setTanggalPengembalian(tglPengembalian);

confirmed = true;
dispose();
}

private void showError(String message) {
    JOptionPane.showMessageDialog(this, "Warning!" + message, "Input Error",
JOptionPane.WARNING_MESSAGE);
}

public boolean isConfirmed() {
    return confirmed;
}

public Peminjaman getPeminjaman() {
    return peminjaman;
}

```

```
}
```

15. Peminjaman Management Panel

```
package com.Library.Admin;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.DefaultTableCellRenderer;
import java.awt.*;
import java.sql.Date;
import java.text.SimpleDateFormat;
import java.util.List;

public class PeminjamanManagementPanel extends JPanel {

    private PeminjamanDAO peminjamanDAO;
    private JTable peminjamanTable;
    private DefaultTableModel tableModel;
    private JTextField searchField;
    private JButton addButton, processReturnButton, refreshButton, searchButton, delleteButton;
    private JLabel statusLabel;
    private SimpleDateFormat dateFormat = new SimpleDateFormat("dd MMM yyyy");
}
```

```

private final Color PRIMARY_GREEN = new Color(34, 139, 34); // Forest Green

private final Color LIGHT_GREEN = new Color(144, 238, 144); // Light Green

private final Color DARK_GREEN = new Color(0, 100, 0); // Dark Green

private final Color ACCENT_GREEN = new Color(50, 205, 50); // Lime Green

private final Color BG_GREEN = new Color(240, 255, 240); // Honeydew


public PeminjamanManagementPanel() {

    peminjamanDAO = new PeminjamanDAO();

    initComponents();

    setupLayout();

    setupEvents();

    loadPeminjamanData();

}

private void initComponents() {

    setBackground(Color.WHITE); // Background utama panel

    // Table

    String[] columns = {"ID Pinjam", "Judul Buku", "Nama Anggota", "Tgl Pinjam", "Tgl Jatuh
Tempo", "Tgl Kembali", "Status"};

    tableModel = new DefaultTableModel(columns, 0) {

```

```

@Override

public boolean isCellEditable(int row, int column) {

    return false;
}

};

peminjamanTable = new JTable(tableModel);

peminjamanTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

peminjamanTable.setRowHeight(30);

peminjamanTable.setFont(new Font("Arial", Font.PLAIN, 12));

peminjamanTable.getTableHeader().setFont(new Font("Arial", Font.BOLD, 13));

peminjamanTable.getTableHeader().setBackground(PRIMARY_GREEN);

peminjamanTable.getTableHeader().setForeground(Color.WHITE);

peminjamanTable.setGridColor(LIGHT_GREEN);

peminjamanTable.setSelectionBackground(LIGHT_GREEN);

peminjamanTable.setSelectionForeground(DARK_GREEN);

// Mengatur lebar kolom

peminjamanTable.getColumnModel().getColumn(0).setPreferredWidth(60); // ID Pinjam

peminjamanTable.getColumnModel().getColumn(1).setPreferredWidth(200); // Judul Buku

peminjamanTable.getColumnModel().getColumn(2).setPreferredWidth(150); // Nama Anggota

```

```

peminjamanTable.getColumnModel().getColumn(3).setPreferredWidth(100); // Tgl Pinjam

peminjamanTable.getColumnModel().getColumn(4).setPreferredWidth(110); // Tgl Jatuh Tempo

peminjamanTable.getColumnModel().getColumn(5).setPreferredWidth(100); // Tgl Kembali

peminjamanTable.getColumnModel().getColumn(6).setPreferredWidth(100); // Status

// Custom renderer untuk kolom Status Peminjaman

        peminjamanTable.getColumnModel().getColumn(6).setCellRenderer(new
DefaultTableCellRenderer() {

    @Override

    public Component getTableCellRendererComponent(JTable table, Object value,
boolean isSelected, boolean hasFocus, int row, int column) {

        Component c = super.getTableCellRendererComponent(table, value, isSelected, hasFocus,
row, column);

        if (!isSelected) {

            String status = (String) value;

            if ("Dipinjam".equalsIgnoreCase(status)) {

                c.setBackground(new Color(255, 255, 220)); // Light Yellow

                c.setForeground(new Color(139, 69, 19)); // SaddleBrown

            } else if ("Dikembalikan".equalsIgnoreCase(status)) {

                c.setBackground(new Color(220, 255, 220)); // Light Green

                c.setForeground(DARK_GREEN);

```

```

} else if ("Terlambat".equalsIgnoreCase(status)) {

    c.setBackground(new Color(255, 200, 200)); // Light Red

    c.setForeground(new Color(139, 0, 0)); // Dark Red

} else {

    c.setBackground(Color.WHITE);

    c.setForeground(Color.BLACK);

}

}

setHorizontalAlignment(SwingConstants.CENTER);

return c;

});

}

// Search field

searchField = new JTextField(20);

searchField.setFont(new Font("Arial", Font.PLAIN, 14));

}

// Status label

statusLabel = new JLabel("Total Peminjaman Aktif: 0");

statusLabel.setFont(new Font("Arial", Font.BOLD, 12));

statusLabel.setForeground(DARK_GREEN);

```

```

// Buttons

addButton = new JButton("Buat Peminjaman");

processReturnButton = new JButton("Proses Pengembalian/Edit");

refreshButton = new JButton("Refresh");

searchButton = new JButton("Cari");

delleteButton = new JButton("Hapus Peminjaman");

styleButton(addButton, ACCENT_GREEN);

styleButton(processReturnButton, new Color(255, 165, 0));

styleButton(refreshButton, PRIMARY_GREEN);

styleButton(searchButton, DARK_GREEN);

styleButton(delleteButton, new Color(139, 0, 0));

}

private void styleButton(JButton button, Color color) {

button.setBackground(color);

button.setForeground(Color.WHITE);

button.setFont(new Font("Arial", Font.BOLD, 12));

button.setPreferredSize(new Dimension(180, 35));

button.setFocusPainted(false);

button.setCursor(new Cursor(Cursor.HAND_CURSOR));

button.setBorder(BorderFactory.createRaisedBevelBorder());

```

```

}

private void setupLayout() {
    setLayout(new BorderLayout(0,0));

    // Header Panel (Judul Panel)
    JPanel headerPanelTitle = new JPanel(new FlowLayout(FlowLayout.CENTER));
    headerPanelTitle.setBackground(DARK_GREEN);
    headerPanelTitle.setPreferredSize(new Dimension(0, 50));

    JLabel titleLabel = new JLabel("MANAGEMENT PEMINJAMAN BUKU");
    titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
    titleLabel.setForeground(Color.WHITE);
    headerPanelTitle.add(titleLabel);

    // Top Panel (Search + Status)
    JPanel topPanel = new JPanel(new BorderLayout(10,10));
    topPanel.setBackground(BG_GREEN);
    topPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

    JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 10, 0));
    searchPanel.setOpaque(false);
}

```

```
JLabel searchLabel = new JLabel("Cari (Judul/Anggota/Status):");

searchLabel.setFont(new Font("Arial", Font.BOLD, 14));

searchLabel.setForeground(DARK_GREEN);

searchPanel.add(searchLabel);

searchPanel.add(searchField);

searchPanel.add(searchButton);

JPanel statusPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT, 0, 0));

statusPanel.setOpaque(false);

statusPanel.add(statusLabel);

topPanel.add(searchPanel, BorderLayout.WEST);

topPanel.add(statusPanel, BorderLayout.EAST);

JPanel northPanelContainer = new JPanel(new BorderLayout());

northPanelContainer.add(headerPanelTitle, BorderLayout.NORTH);

northPanelContainer.add(topPanel, BorderLayout.CENTER);

add(northPanelContainer, BorderLayout.NORTH);

// Table Panel

JScrollPane scrollPane = new JScrollPane(peminjamanTable);
```

```

scrollPane.setBorder(BorderFactory.createTitledBorder(
    BorderFactory.createLineBorder(PRIMARY_GREEN, 2),
    "Daftar Transaksi Peminjaman", 0, 0,
    new Font("Arial", Font.BOLD, 14), PRIMARY_GREEN));
scrollPane.getViewport().setBackground(Color.WHITE);
add(scrollPane, BorderLayout.CENTER);

// Button Panel
JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 15, 15));
buttonPanel.setBackground(BG_GREEN); // Background untuk area tombol
buttonPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
buttonPanel.add(addButton);
buttonPanel.add(processReturnButton);
buttonPanel.add(deleteButton);
buttonPanel.add(refreshButton);
add(buttonPanel, BorderLayout.SOUTH);
}

private void setupEvents() {
    addButton.addActionListener(e -> showPeminjamanDialog(null));
    processReturnButton.addActionListener(e -> showPeminjamanDialogForUpdate());
    refreshButton.addActionListener(e -> loadPeminjamanData());
}

```

```

searchButton.addActionListener(e -> searchPeminjamanData());

searchField.addActionListener(e -> searchPeminjamanData());

delleteButton.addActionListener(e -> deletePeminjamanData());

}

private String formatDate(Date date) {

    if (date == null) {

        return "-";

    }

    return dateFormat.format(date);

}

private void loadPeminjamanData() {

    SwingUtilities.invokeLater(() -> {

        try {

            tableModel.setRowCount(0); // Clear table

            List<Peminjaman> daftarPeminjaman = peminjamanDAO.getAllPeminjaman();

        }

    });

    for (Peminjaman p : daftarPeminjaman) {

        Object[] row = {

            p.getIdPeminjaman(),

            p.getJudulBuku(),

```

```

        p.getNamaAnggota(),
        formatDate(p.getTanggalPinjam()),
        formatDate(p.getTanggalJatuhTempo()),
        formatDate(p.getTanggalPengembalian()),
        p.getStatusPeminjaman()
    };

    tableModel.addRow(row);
}

statusLabel.setText("Total Peminjaman Aktif: " +
peminjamanDAO.getTotalPeminjamanAktif());

} catch (Exception e) {
    JOptionPane.showMessageDialog(this,
    "Error loading peminjaman data: " + e.getMessage(),
    "Error", JOptionPane.ERROR_MESSAGE);

    e.printStackTrace();
}
});

}

private void searchPeminjamanData() {
    String keyword = searchField.getText().trim();

    if(keyword.isEmpty()) {
        loadPeminjamanData();
    }
}

```

```

        return;
    }

    SwingUtilities.invokeLater(() -> {
        try {
            tableModel.setRowCount(0);

            List<Peminjaman> results = peminjamanDAO.searchPeminjaman(keyword);

            for (Peminjaman p : results) {
                Object[] row = {
                    p.getIdPeminjaman(),
                    p.getJudulBuku(),
                    p.getNamaAnggota(),
                    formatDate(p.getTanggalPinjam()),
                    formatDate(p.getTanggalJatuhTempo()),
                    formatDate(p.getTanggalPengembalian()),
                    p.getStatusPeminjaman()
                };

                tableModel.addRow(row);
            }
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this,
                "Error searching peminjaman: " + e.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    });
}

```

```

        "Error", JOptionPane.ERROR_MESSAGE);

    }

});

}

private void showPeminjamanDialog(Peminjaman peminjamanToEdit) {
    Frame parentFrame = (Frame) SwingUtilities.getWindowAncestor(this);

    String dialogTitle = (peminjamanToEdit == null) ? "Tambah Peminjaman Baru" : "Edit Data
Peminjaman";

    PeminjamanDialog dialog = new PeminjamanDialog(parentFrame, dialogTitle,
peminjamanToEdit);

    dialog.setVisible(true);

}

if (dialog.isConfirmed()) {

    Peminjaman peminjamanData = dialog.getPeminjaman();

    boolean success;

    if (peminjamanToEdit == null) { // Tambah baru

        success = peminjamanDAO.addPeminjaman(peminjamanData);

        if (success) {

            JOptionPane.showMessageDialog(this, "Peminjaman berhasil ditambahkan!", "Sukses",
JOptionPane.INFORMATION_MESSAGE);

        } else {

            JOptionPane.showMessageDialog(this, "Gagal menambahkan peminjaman!", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

```
    }

} else { // Edit

    peminjamanData.setIdPeminjaman(peminjamanToEdit.getIdPeminjaman()); // Pastikan ID
ter-set

    success = peminjamanDAO.updatePeminjaman(peminjamanData);

    if(success) {

        JOptionPane.showMessageDialog(this, "Data peminjaman berhasil diupdate!", "Sukses",
JOptionPane.INFORMATION_MESSAGE);

    } else {

        JOptionPane.showMessageDialog(this, "Gagal mengupdate data peminjaman!", "Error",
JOptionPane.ERROR_MESSAGE);

    }
}

if(success) loadPeminjamanData();

}

}

private void showPeminjamanDialogForUpdate() {

    int selectedRow = peminjamanTable.getSelectedRow();

    if(selectedRow == -1) {

        JOptionPane.showMessageDialog(this,
        "Pilih transaksi peminjaman yang ingin diproses/diedit!",
        "Peringatan", JOptionPane.WARNING_MESSAGE);

    }

    return;
}
```

```

    }

int idPeminjaman = (Integer) tableModel.getValueAt(selectedRow, 0);

Peminjaman selectedPeminjaman = new Peminjaman();

selectedPeminjaman.setIdPeminjaman((Integer) tableModel.getValueAt(selectedRow, 0));

selectedPeminjaman.setJudulBuku((String) tableModel.getValueAt(selectedRow, 1));

selectedPeminjaman.setNamaAnggota((String) tableModel.getValueAt(selectedRow, 2));

List<Peminjaman> currentList = peminjamanDAO.getAllPeminjaman();

Peminjaman actualPeminjamanToEdit = null;

for(Peminjaman p : currentList) {

    if(p.getIdPeminjaman() == idPeminjaman) {

        actualPeminjamanToEdit = p;

        break;
    }
}

if(actualPeminjamanToEdit == null) {

    JOptionPane.showMessageDialog(this, "Data peminjaman tidak ditemukan untuk diedit.", "Error", JOptionPane.ERROR_MESSAGE);
}

return;
}

```

```

        showPeminjamanDialog(actualPeminjamanToEdit);

    }

private void deletePeminjamanData() {
    int selectedRow = peminjamanTable.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(this,
            "Pilih data peminjaman yang akan dihapus!",
            "Peringatan",
            JOptionPane.WARNING_MESSAGE);
    }
    return;
}

int idPeminjaman = (Integer) tableModel.getValueAt(selectedRow, 0);
String judulBuku = (String) tableModel.getValueAt(selectedRow, 1);
String namaAnggota = (String) tableModel.getValueAt(selectedRow, 2);

int confirm = JOptionPane.showConfirmDialog(this,
    "Apakah Anda yakin ingin menghapus data peminjaman:\n" + namaAnggota + " (judul buku: "
    + judulBuku + ")?",

    "Konfirmasi Hapus",
    JOptionPane.YES_NO_OPTION,

```

```
JOptionPane.QUESTION_MESSAGE);  
  
if(confirm == JOptionPane.YES_OPTION) {  
  
    if(peminjamanDAO.deletePeminjaman(idPeminjaman)) {  
  
        JOptionPane.showMessageDialog(this,  
            "Data berhasil dihapus!",  
            "Sukses",  
            JOptionPane.INFORMATION_MESSAGE);  
  
        loadPeminjamanData();  
  
    } else {  
  
        JOptionPane.showMessageDialog(this,  
            "Gagal menghapus data!",  
            "Error",  
            JOptionPane.ERROR_MESSAGE);  
  
    }  
}  
}  
}  
}
```

16. Pengumuman

```
package com.Library.Admin;

import java.sql.Date;

public class Pengumuman {

    private int idPengumuman;
    private String judul;
    private String isi;
    private Date tanggalDibuat;

    public Pengumuman() {
    }

    public Pengumuman(int idPengumuman, String judul, String isi, Date tanggalDibuat) {
        this.idPengumuman = idPengumuman;
        this.judul = judul;
        this.isi = isi;
        this.tanggalDibuat = tanggalDibuat;
    }

    public int getIdPengumuman() {
```

```
return idPengumuman;

}

public void setIdPengumuman(int idPengumuman) {

    this.idPengumuman = idPengumuman;

}

public String getJudul() {

    return judul;

}

public void setJudul(String judul) {

    this.judul = judul;

}

public String getIsi() {

    return isi;

}

public void setIsi(String isi) {

    this.isi = isi;

}
```

```
public Date getTanggalDibuat() {  
    return tanggalDibuat;  
}  
  
public void setTanggalDibuat(Date tanggalDibuat) {  
    this.tanggalDibuat = tanggalDibuat;  
}  
  
@Override  
public String toString() {  
    return judul + " (ID: " + idPengumuman + ")";  
}  
}
```

17. Pengumuman DAO (Data Access Object)

```
package com.Library.Admin;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;
```

```

import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Date;
import java.util.ArrayList;
import java.util.List;

public class PengumumanDAO {
    private Connection connection;

    public PengumumanDAO() {
        this.connection = DBConnection.getConnection();
        if (this.connection == null) {
            System.err.println("Koneksi database gagal didapatkan di PengumumanDAO.");
        }
    }

    public boolean addPengumuman(Pengumuman pengumuman) {
        if (connection == null) return false; // Tidak bisa operasi tanpa koneksi
        String sql = "INSERT INTO pengumuman (judul, isi, tanggal_dibuat) VALUES (?, ?, ?)";
        try (PreparedStatement pstmt = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS)) {
            pstmt.setString(1, pengumuman.getJudul());

```

```

        pstmt.setString(2, pengumuman.getIsi());

        // Jika tanggal dibuat null, set ke tanggal sekarang

        if (pengumuman.getTanggalDibuat() == null) {

            pengumuman.setTanggalDibuat(new Date(System.currentTimeMillis()));

        }

        pstmt.setDate(3, pengumuman.getTanggalDibuat());

    }

    int affectedRows = pstmt.executeUpdate();

    if (affectedRows > 0) {

        try (ResultSet generatedKeys = pstmt.getGeneratedKeys()) {

            if (generatedKeys.next()) {

                pengumuman.setIdPengumuman(generatedKeys.getInt(1));

            }

        }

        return true;

    }

} catch (SQLException e) {

    System.err.println("SQL Error saat menambah pengumuman: " + e.getMessage());

    e.printStackTrace();

}

return false;

}

```

```

public List<Pengumuman> getAllPengumuman() {
    List<Pengumuman> daftarPengumuman = new ArrayList<>();
    if (connection == null) return daftarPengumuman;

    String sql = "SELECT id_pengumuman, judul, isi, tanggal_dibuat FROM pengumuman ORDER BY tanggal_dibuat DESC, id_pengumuman DESC";
    try (Statement stmt = connection.createStatement());
        ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                Pengumuman p = new Pengumuman();
                p.setIdPengumuman(rs.getInt("id_pengumuman"));
                p.setJudul(rs.getString("judul"));
                p.setIsi(rs.getString("isi"));
                p.setTanggalDibuat(rs.getDate("tanggal_dibuat"));
                daftarPengumuman.add(p);
            }
        } catch (SQLException e) {
            System.err.println("SQL Error saat mengambil semua pengumuman: " + e.getMessage());
            e.printStackTrace();
        }
    return daftarPengumuman;
}

```

```

public Pengumuman getPengumumanById(int id) {

    if (connection == null) return null;

    String sql = "SELECT id_pengumuman, judul, isi, tanggal_dibuat FROM pengumuman WHERE
id_pengumuman = ?";

    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

        pstmt.setInt(1, id);

        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {

            Pengumuman p = new Pengumuman();

            p.setIdPengumuman(rs.getInt("id_pengumuman"));

            p.setJudul(rs.getString("judul"));

            p.setIsi(rs.getString("isi"));

            p.setTanggalDibuat(rs.getDate("tanggal_dibuat"));

            return p;
        }
    }

    rs.close();
}

} catch (SQLException e) {

    System.err.println("SQL Error saat mengambil pengumuman by ID: " + e.getMessage());

    e.printStackTrace();
}

return null;
}

```

```

public boolean updatePengumuman(Pengumuman pengumuman) {
    if (connection == null) return false;

    String sql = "UPDATE pengumuman SET judul = ?, isi = ?, tanggal_dibuat = ? WHERE
id_pengumuman = ?";
    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
        pstmt.setString(1, pengumuman.getJudul());
        pstmt.setString(2, pengumuman.getIsi());
        if (pengumuman.getTanggalDibuat() == null) {
            pengumuman.setTanggalDibuat(new Date(System.currentTimeMillis()));
        }
        pstmt.setDate(3, pengumuman.getTanggalDibuat());
        pstmt.setInt(4, pengumuman.getIdPengumuman());
        int affectedRows = pstmt.executeUpdate();
        return affectedRows > 0;
    } catch (SQLException e) {
        System.err.println("SQL Error saat update pengumuman: " + e.getMessage());
        e.printStackTrace();
    }
    return false;
}

```

```

public boolean deletePengumuman(int idPengumuman) {

    if (connection == null) return false;

    String sql = "DELETE FROM pengumuman WHERE id_pengumuman = ?";

    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

        pstmt.setInt(1, idPengumuman);

        int affectedRows = pstmt.executeUpdate();

        return affectedRows > 0;

    } catch (SQLException e) {

        System.err.println("SQL Error saat menghapus pengumuman: " + e.getMessage());

        e.printStackTrace();

    }

    return false;

}

public List<Pengumuman> searchPengumuman(String keyword) {

    List<Pengumuman> daftarPengumuman = new ArrayList<>();

    if (connection == null) return daftarPengumuman;

    String sql = "SELECT id_pengumuman, judul, isi, tanggal_dibuat FROM pengumuman " +

        "WHERE judul LIKE ? OR isi LIKE ? // Kriteria pencarian pengumuman

        "ORDER BY tanggal_dibuat DESC, id_pengumuman DESC";

    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

        String searchPattern = "%" + keyword + "%";

```

```
pstmt.setString(1, searchPattern);

pstmt.setString(2, searchPattern);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

    Pengumuman p = new Pengumuman();

    p.setIdPengumuman(rs.getInt("id_pengumuman"));

    p.setJudul(rs.getString("judul"));

    p.setIsi(rs.getString("isi"));

    p.setTanggalDibuat(rs.getDate("tanggal_dibuat"));

    daftarPengumuman.add(p);

}

rs.close();

} catch (SQLException e) {

    System.err.println("SQL Error saat mencari pengumuman: " + e.getMessage());

    e.printStackTrace();

}

return daftarPengumuman;

}
```

18. Pengumuman Dialog

```
package com.Library.Admin;

import javax.swing.*;
import java.awt.*;
import java.sql.Date;
import java.text.ParseException;
import java.text.SimpleDateFormat;

public class PengumumanDialog extends JDialog {

    private JTextField judulField;
    private JTextArea isiArea;
    private JTextField tanggalField;
    private JButton saveButton;
    private JButton cancelButton;
    private boolean confirmed = false;
    private Pengumuman pengumuman;
    private SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

    private final Color PRIMARY_GREEN = new Color(34, 139, 34);
```

```

private final Color DARK_GREEN = new Color(0, 100, 0);

private final Color BG_GREEN = new Color(240, 255, 240);

public PengumumanDialog(Frame parent, String dialogTitle, Pengumuman pengumumanToEdit) {
    super(parent, dialogTitle, true);
    this.pengumuman = pengumumanToEdit;
}

initComponents();

setupLayout();

setupEvents();

if (pengumumanToEdit != null) {
    populateFields(pengumumanToEdit);
} else {
    tanggalField.setText(dateFormat.format(new java.util.Date())); // Tanggal hari ini untuk baru
}

}

private void initComponents() {
    setSize(600, 500);
    setLocationRelativeTo(getParent());
    setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
}

```

```

judulField = new JTextField(30);

isiArea = new JTextArea(10, 30);

isiArea.setLineWrap(true);

isiArea.setWrapStyleWord(true);

tanggalField = new JTextField(10);

saveButton = new JButton("Simpan");

cancelButton = new JButton("Batal");

styleTextField(judulField);

styleTextArea(isiArea);

styleTextField(tanggalField);

styleButton(saveButton, PRIMARY_GREEN);

styleButton(cancelButton, new Color(220, 20, 60));

}

private void styleTextField(JTextField field) {

    field.setFont(new Font("Arial", Font.PLAIN, 14));

    field.setBorder(BorderFactory.createCompoundBorder(
        BorderFactory.createLineBorder(PRIMARY_GREEN, 1),
        BorderFactory.createEmptyBorder(5, 8, 5, 8)
    ));
}

```

```
));
}

private void styleTextArea(JTextArea area) {
    area.setFont(new Font("Arial", Font.PLAIN, 14));
    area.setBorder(BorderFactory.createCompoundBorder(
        BorderFactory.createLineBorder(PRIMARY_GREEN, 1),
        BorderFactory.createEmptyBorder(5, 8, 5, 8)
    ));
}

private void styleButton(JButton button, Color color) {
    button.setBackground(color);
    button.setForeground(Color.WHITE);
    button.setFont(new Font("Arial", Font.BOLD, 14));
    button.setPreferredSize(new Dimension(120, 40));
    button.setFocusPainted(false);
    button.setCursor(new Cursor(Cursor.HAND_CURSOR));
    button.setBorder(BorderFactory.createRaisedBevelBorder());
}

private void setupLayout() {
```

```

setLayout(new BorderLayout(10, 10));

getContentPane().setBackground(BG_GREEN);

JPanel headerPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));

headerPanel.setBackground(DARK_GREEN);

headerPanel.setPreferredSize(new Dimension(0, 50));

JLabel titleLabelDialog = new JLabel(getTitle());

titleLabelDialog.setFont(new Font("Arial", Font.BOLD, 18));

titleLabelDialog.setForeground(Color.WHITE);

headerPanel.add(titleLabelDialog);

add(headerPanel, BorderLayout.NORTH);

JPanel formPanel = new JPanel(new GridBagLayout());

formPanel.setBackground(BG_GREEN);

formPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(8, 8, 8, 8);

gbc.anchor = GridBagConstraints.WEST;

gbc.gridx = 0; gbc.gridy = 0;

JLabel judulLabel = new JLabel("Judul:");

styleFormLabel(judulLabel);

```

```

formPanel.add(judulLabel, gbc);

gbc.gridx = 1; gbc.gridy = 0; gbc.gridwidth = 2; gbc.fill = GridBagConstraints.HORIZONTAL;

formPanel.add(judulField, gbc);

gbc.gridx = 0; gbc.gridy = 1; gbc.gridwidth = 1; gbc.fill = GridBagConstraints.NONE;

JLabel isiLabel = new JLabel("Isi Pengumuman:");

styleFormLabel(isiLabel);

formPanel.add(isiLabel, gbc);

gbc.gridx = 1; gbc.gridy = 1; gbc.gridwidth = 2; gbc.fill = GridBagConstraints.BOTH;
gbc.weightx = 1.0; gbc.weighty = 1.0;

formPanel.add(new JScrollPane(isiArea), gbc);

gbc.gridx = 0; gbc.gridy = 2; gbc.gridwidth = 1; gbc.fill = GridBagConstraints.NONE;
gbc.weighty = 0;

JLabel tanggalLabel = new JLabel("Tanggal (YYYY-MM-DD):");

styleFormLabel(tanggalLabel);

formPanel.add(tanggalLabel, gbc);

gbc.gridx = 1; gbc.gridy = 2; gbc.gridwidth = 1; gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.weightx = 0;

formPanel.add(tanggalField, gbc);

add(formPanel, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 15, 10));

```

```

buttonPanel.setBackground(BG_GREEN);

buttonPanel.setBorder(BorderFactory.createEmptyBorder(0,0,10,0));

buttonPanel.add(saveButton);

buttonPanel.add(cancelButton);

add(buttonPanel, BorderLayout.SOUTH);

}

private void styleFormLabel(JLabel label) {

label.setFont(new Font("Arial", Font.BOLD, 14));

label.setForeground(DARK_GREEN);

}

private void setupEvents() {

saveButton.addActionListener(e -> savePengumuman());

cancelButton.addActionListener(e -> {

confirmed = false;

dispose();

});

}

private void populateFields(Pengumuman p) {

judulField.setText(p.getJudul());

```

```

isiArea.setText(p.getIsi());

tanggalField.setText(p.getTanggalDibuat() != null ? dateFormat.format(p.getTanggalDibuat()) : "");
}

private Date parseDate(String dateStr) {

if (dateStr == null || dateStr.trim().isEmpty()) {

showError("Tanggal tidak boleh kosong.");

return null;
}

try {

java.util.Date utilDate = dateFormat.parse(dateStr.trim());

return new Date(utilDate.getTime());
} catch (ParseException e) {

showError("Format tanggal salah. Gunakan YYYY-MM-DD (Contoh: 2024-01-15).");

return null;
}
}

private void savePengumuman() {

String judul = judulField.getText().trim();

String isi = isiArea.getText().trim();

String tanggalStr = tanggalField.getText().trim();

```

```

if (judul.isEmpty()) {
    showError("Judul pengumuman tidak boleh kosong!");
    judulField.requestFocus();
    return;
}

if (isi.isEmpty()) {
    showError("Isi pengumuman tidak boleh kosong!");
    isiArea.requestFocus();
    return;
}

Date tanggalDibuat = parseDate(tanggalStr);
if (tanggalDibuat == null) {
    tanggalField.requestFocus();
    return;
}

if (this.pengumuman == null) {
    this.pengumuman = new Pengumuman();
}

```

```
this.pengumuman.setJudul(judul);

this.pengumuman.setIsi(isi);

this.pengumuman.setTanggalDibuat(tanggalDibuat);

confirmed = true;

dispose();

}

private void showError(String message) {

    JOptionPane.showMessageDialog(this, message, "Input Error",

JOptionPane.ERROR_MESSAGE);

}

public boolean isConfirmed() {

    return confirmed;

}

public Pengumuman getPengumuman() {

    return pengumuman;

}

}
```

19. Pengumuman Management Panel

```
package com.Library.Admin;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.DefaultTableCellRenderer;
import java.awt.*;
import java.sql.Date;
import java.text.SimpleDateFormat;
import java.util.List;

public class PengumumanManagementPanel extends JPanel {

    private PengumumanDAO pengumumanDAO;
    private JTable pengumumanTable;
    private DefaultTableModel tableModel;
    private JTextField searchField;
    private JButton addButton, editButton, deleteButton, refreshButton, searchButton;
    private JLabel statusLabel;
    private SimpleDateFormat tableDateFormat = new SimpleDateFormat("dd MMM yyyy");

    private final Color PRIMARY_GREEN = new Color(34, 139, 34);
```

```

private final Color LIGHT_GREEN = new Color(144, 238, 144);

private final Color DARK_GREEN = new Color(0, 100, 0);

private final Color ACCENT_GREEN = new Color(50, 205, 50);

private final Color BG_GREEN = new Color(240, 255, 240);

public PengumumanManagementPanel() {

    pengumumanDAO = new PengumumanDAO();

    initComponents();

    setupLayout();

    setupEvents();

    loadPengumumanData();

}

private void initComponents() {

    setBackground(Color.WHITE);

}

String[] columns = {"ID", "Judul", "Isi (Ringkas)", "Tanggal"};

tableModel = new DefaultTableModel(columns, 0) {

    @Override

    public boolean isCellEditable(int row, int column) {

        return false;

    }

}

```

```

};

pengumumanTable = new JTable(tableModel);

pengumumanTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

pengumumanTable.setRowHeight(28);

pengumumanTable.setFont(new Font("Arial", Font.PLAIN, 12));

pengumumanTable.getTableHeader().setFont(new Font("Arial", Font.BOLD, 13));

pengumumanTable.getTableHeader().setBackground(PRIMARY_GREEN);

pengumumanTable.getTableHeader().setForeground(Color.WHITE);

pengumumanTable.setGridColor(LIGHT_GREEN);

pengumumanTable.setSelectionBackground(LIGHT_GREEN);

pengumumanTable.setSelectionForeground(DARK_GREEN);

pengumumanTable.getColumnModel().getColumn(0).setPreferredWidth(40);

pengumumanTable.getColumnModel().getColumn(1).setPreferredWidth(250);

pengumumanTable.getColumnModel().getColumn(2).setPreferredWidth(350);

pengumumanTable.getColumnModel().getColumn(3).setPreferredWidth(100);

pengumumanTable.getColumnModel().getColumn(3).setCellRenderer(new
DefaultTableCellRenderer() {

@Override

public Component getTableCellRendererComponent(JTable table, Object value,
boolean isSelected, boolean hasFocus, int row, int column) {

```

```

        if (value instanceof Date) {

            value = tableDateFormat.format((Date) value);

        }

        return super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row,
column);

    });

    pengumumanTable.getColumnModel().getColumn(2).setCellRenderer(new
DefaultTableCellRenderer() {

    @Override

    public Component getTableCellRendererComponent(JTable table, Object value,

        boolean isSelected, boolean hasFocus, int row, int column) {

        super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);

        if (value instanceof String) {

            String text = (String) value;

            setText(text.length() > 80 ? text.substring(0, 77) + "..." : text);

        }

        return this;

    }

});

searchField = new JTextField(20);

searchField.setFont(new Font("Arial", Font.PLAIN, 14));

```

```

statusLabel = new JLabel("Total: 0 pengumuman");

statusLabel.setFont(new Font("Arial", Font.BOLD, 12));

statusLabel.setForeground(DARK_GREEN);

addButton = new JButton("Tambah Pengumuman");

editButton = new JButton("Edit Pengumuman");

deleteButton = new JButton("Hapus Pengumuman");

refreshButton = new JButton("Refresh");

searchButton = new JButton("Cari");

styleButton(addButton, ACCENT_GREEN, 180);

styleButton(editButton, new Color(255, 165, 0), 180);

styleButton(deleteButton, new Color(139, 0, 0), 180);

styleButton(refreshButton, PRIMARY_GREEN, 120);

styleButton(searchButton, DARK_GREEN, 100);

}

private void styleButton(JButton button, Color color, int width) {

button.setBackground(color);

button.setForeground(Color.WHITE);

button.setFont(new Font("Arial", Font.BOLD, 12));

button.setPreferredSize(new Dimension(width, 35));

```

```

button.setFocusPainted(false);

button.setCursor(new Cursor(Cursor.HAND_CURSOR));

button.setBorder(BorderFactory.createRaisedBevelBorder());

}

private void setupLayout() {

setLayout(new BorderLayout(0,0));

JPanel headerPanelTitle = new JPanel(new FlowLayout(FlowLayout.CENTER));

headerPanelTitle.setBackground(DARK_GREEN);

headerPanelTitle.setPreferredSize(new Dimension(0, 50));

JLabel titleLabel = new JLabel("MANAGEMENT PENGUMUMAN");

titleLabel.setFont(new Font("Arial", Font.BOLD, 20));

titleLabel.setForeground(Color.WHITE);

headerPanelTitle.add(titleLabel);

JPanel topPanel = new JPanel(new BorderLayout(10,10));

topPanel.setBackground(BG_GREEN);

topPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 10, 0));

searchPanel.setOpaque(false);

```

```

JLabel searchLabel = new JLabel("Cari (Judul/Isi):");

searchLabel.setFont(new Font("Arial", Font.BOLD, 14));

searchLabel.setForeground(DARK_GREEN);

searchPanel.add(searchLabel);

searchPanel.add(searchField);

searchPanel.add(searchButton);

JPanel statusPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT, 0, 0));

statusPanel.setOpaque(false);

statusPanel.add(statusLabel);

topPanel.add(searchPanel, BorderLayout.WEST);

topPanel.add(statusPanel, BorderLayout.EAST);

JPanel northPanelContainer = new JPanel(new BorderLayout());

northPanelContainer.add(headerPanelTitle, BorderLayout.NORTH);

northPanelContainer.add(topPanel, BorderLayout.CENTER);

add(northPanelContainer, BorderLayout.NORTH);

JScrollPane scrollPane = new JScrollPane(pengumumanTable);

scrollPane.setBorder(BorderFactory.createTitledBorder(
    BorderFactory.createLineBorder(PRIMARY_GREEN, 2),

```

```

        "Daftar Pengumuman", 0, 0,
        new Font("Arial", Font.BOLD, 14), PRIMARY_GREEN));
scrollPane.setViewport().setBackground(Color.WHITE);
add(scrollPane, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 15, 15));
buttonPanel.setBackground(BG_GREEN);
buttonPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
buttonPanel.add(addButton);
buttonPanel.add(editButton);
buttonPanel.add(deleteButton);
buttonPanel.add(refreshButton);
add(buttonPanel, BorderLayout.SOUTH);
}

private void setupEvents() {
    addButton.addActionListener(e -> showPengumumanDialog(null, "Tambah Pengumuman Baru"));
    editButton.addActionListener(e -> editSelectedPengumuman());
    deleteButton.addActionListener(e -> deleteSelectedPengumuman());
    refreshButton.addActionListener(e -> loadPengumumanData());
    searchButton.addActionListener(e -> searchPengumumanData());
    searchField.addActionListener(e -> searchPengumumanData());
}

```

```

}

private void loadPengumumanData() {
    SwingUtilities.invokeLater(() -> {
        try {
            tableModel.setRowCount(0);
            List<Pengumuman> daftarPengumuman = pengumumanDAO.getAllPengumuman();
            for (Pengumuman p : daftarPengumuman) {
                Object[] row = {
                    p.getIdPengumuman(),
                    p.getJudul(),
                    p.getIsi(),
                    p.getTanggalDibuat()
                };
                tableModel.addRow(row);
            }
            statusLabel.setText("Total: " + daftarPengumuman.size() + " pengumuman");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Error memuat data: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
        }
    });
}

```

```

private void searchPengumumanData() {

    String keyword = searchField.getText().trim();

    SwingUtilities.invokeLater(() -> {

        try {

            tableModel.setRowCount(0);

            List<Pengumuman> results = keyword.isEmpty() ? pengumumanDAO.getAllPengumuman()
                : pengumumanDAO.searchPengumuman(keyword);

            for (Pengumuman p : results) {

                Object[] row = { p.getIdPengumuman(), p.getJudul(), p.getIsi(), p.getTanggalDibuat() };

                tableModel.addRow(row);

            }

            statusLabel.setText((keyword.isEmpty() ? "Total: " : "Hasil Pencarian: ") + results.size() + "
pengumuman");

        } catch (Exception e) {

            JOptionPane.showMessageDialog(this, "Error mencari data: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);

        }

    });

}

private void showPengumumanDialog(Pengumuman pengumumanToEdit, String dialogTitle) {

    Frame parentFrame = (Frame) SwingUtilities.getWindowAncestor(this);

```

```

        PengumumanDialog dialog = new PengumumanDialog(parentFrame, dialogTitle,
pengumumanToEdit);

        dialog.setVisible(true);

        if (dialog.isConfirmed()) {

            Pengumuman pengumumanData = dialog.getPengumuman();

            boolean success;

            if (pengumumanToEdit == null) {

                success = pengumumanDAO.addPengumuman(pengumumanData);

                if (success) JOptionPane.showMessageDialog(this, "Pengumuman berhasil ditambahkan!",
"Sukses", JOptionPane.INFORMATION_MESSAGE);

                else JOptionPane.showMessageDialog(this, "Gagal menambahkan pengumuman!", "Error",
JOptionPane.ERROR_MESSAGE);

            } else {

                success = pengumumanDAO.updatePengumuman(pengumumanData);

                if (success) JOptionPane.showMessageDialog(this, "Pengumuman berhasil diupdate!",
"Sukses", JOptionPane.INFORMATION_MESSAGE);

                else JOptionPane.showMessageDialog(this, "Gagal mengupdate pengumuman!", "Error",
JOptionPane.ERROR_MESSAGE);

            }

            if (success) loadPengumumanData();

        }

    }

    private void editSelectedPengumuman() {

```

```

int selectedRow = pengumumanTable.getSelectedRow();

if(selectedRow == -1) {

    JOptionPane.showMessageDialog(this, "Pilih pengumuman yang ingin diedit!", "Peringatan",
JOptionPane.WARNING_MESSAGE);

    return;
}

int idPengumuman = (Integer) tableModel.getValueAt(selectedRow, 0);

Pengumuman pengumumanToEdit = pengumumanDAO.getPengumumanById(idPengumuman);
// Ambil data lengkap dari DAO

}

if(pengumumanToEdit == null) {

    JOptionPane.showMessageDialog(this, "Data pengumuman tidak ditemukan untuk diedit.",
"Error", JOptionPane.ERROR_MESSAGE);

    return;
}

showPengumumanDialog(pengumumanToEdit, "Edit Pengumuman");

}

private void deleteSelectedPengumuman() {

int selectedRow = pengumumanTable.getSelectedRow();

if(selectedRow == -1) {

    JOptionPane.showMessageDialog(this, "Pilih pengumuman yang ingin dihapus!",
"Peringatan", JOptionPane.WARNING_MESSAGE);

    return;
}

```

```

    }

    int idPengumuman = (Integer) tableModel.getValueAt(selectedRow, 0);

    String judul = (String) tableModel.getValueAt(selectedRow, 1);

    int confirm = JOptionPane.showConfirmDialog(this,
        "Yakin ingin menghapus pengumuman:\n\"" + judul + "\"?", "Konfirmasi Hapus",
        JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);

    if(confirm == JOptionPane.YES_OPTION) {
        if(pengumumanDAO.deletePengumuman(idPengumuman)) {
            JOptionPane.showMessageDialog(this, "Pengumuman berhasil dihapus!", "Sukses",
                JOptionPane.INFORMATION_MESSAGE);
            loadPengumumanData();
        } else {
            JOptionPane.showMessageDialog(this, "Gagal menghapus pengumuman!", "Error",
                JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

20. Pengunjung

```
package com.Library.Admin;

import java.sql.Date;

public class Pengunjung {

    private int idPengunjung;
    private String nama;
    private String noTelepon;
    private Date tanggalKunjungan;

    // Default constructor

    public Pengunjung() {
    }

    // Constructor dengan parameter

    public Pengunjung(int idPengunjung, String nama, String noTelepon, Date tanggalKunjungan) {
        this.idPengunjung = idPengunjung;
        this.nama = nama;
        this.noTelepon = noTelepon;
        this.tanggalKunjungan = tanggalKunjungan;
    }
}
```

```
// Getter dan Setter

public int getIdPengunjung() {

    return idPengunjung;

}

public void setIdPengunjung(int idPengunjung) {

    this.idPengunjung = idPengunjung;

}

public String getNama() {

    return nama;

}

public void setNama(String nama) {

    this.nama = nama;

}

public String getNoTelepon() {

    return noTelepon;

}
```

```

public void setNoTelepon(String noTelepon) {
    this.noTelepon = noTelepon;
}

public Date getTanggalKunjungan() {
    return tanggalKunjungan;
}

public void setTanggalKunjungan(Date tanggalKunjungan) {
    this.tanggalKunjungan = tanggalKunjungan;
}

@Override
public String toString() {
    return "Pengunjung{" +
        "idPengunjung=" + idPengunjung +
        ", nama='" + nama + '\'' +
        ", noTelepon='" + noTelepon + '\'' +
        ", tanggalKunjungan=" + tanggalKunjungan +
        '}';
}
}

```

21. Pengunjung DAO (Data Access Object)

```
package com.Library.Admin;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PengunjungDAO {

    // Method untuk mengambil semua data pengunjung

    public List<Pengunjung> getAllPengunjung() throws SQLException {
        List<Pengunjung> pengunjungList = new ArrayList<>();

        String query = "SELECT id_pengunjung, nama, no_telepon, tanggal_kunjungan FROM pengunjung ORDER BY tanggal_kunjungan DESC";

        try (Connection connection = DBConnection.getConnection()) {
            PreparedStatement statement = connection.prepareStatement(query);

            ResultSet resultSet = statement.executeQuery();
            while (resultSet.next()) {
                Pengunjung pengunjung = new Pengunjung(
                    resultSet.getInt("id_pengunjung"),
                    resultSet.getString("nama"),
                    resultSet.getString("no_telepon"),
                    resultSet.getDate("tanggal_kunjungan")
                );
                pengunjungList.add(pengunjung);
            }
        }
    }
}
```

```

        resultSet.getInt("id_pengunjung"),
        resultSet.getString("nama"),
        resultSet.getString("no_telepon"),
        resultSet.getDate("tanggal_kunjungan")
    );
    pengunjungList.add(pengunjung);
}
}

return pengunjungList;
}

// Method untuk mencari pengunjung berdasarkan tanggal

public List<Pengunjung> getPengunjungByDate(String dateText) throws SQLException {
    List<Pengunjung> pengunjungList = new ArrayList<>();
    String query = "SELECT id_pengunjung, nama, no_telepon, tanggal_kunjungan FROM pengunjung WHERE DATE(tanggal_kunjungan) = ? ORDER BY tanggal_kunjungan DESC";
}

try (Connection connection = DBConnection.getConnection()) {
    PreparedStatement statement = connection.prepareStatement(query)) {
}

// Convert string to SQL Date
java.sql.Date sqlDate = java.sql.Date.valueOf(dateText);

```

```
statement.setDate(1, sqlDate);

try (ResultSet resultSet = statement.executeQuery()) {

    while (resultSet.next()) {

        Pengunjung pengunjung = new Pengunjung(
            resultSet.getInt("id_pengunjung"),
            resultSet.getString("nama"),
            resultSet.getString("no_telepon"),
            resultSet.getDate("tanggal_kunjungan")
        );

        pengunjungList.add(pengunjung);
    }
}

return pengunjungList;
}
```

22. Pengunjung Management Panel

```
package com.Library.Admin;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.util.List;

public class PengunjungManagementPanel extends JPanel {

    // Color palette

    private final Color PRIMARY_GREEN = new Color(34, 139, 34);    // Forest Green

    private final Color LIGHT_GREEN = new Color(144, 238, 144);    // Light Green

    private final Color DARK_GREEN = new Color(0, 100, 0);        // Dark Green

    private final Color BG_GREEN = new Color(240, 255, 240);      // Honeydew

    private JTable pengunjungTable;
```

```

private DefaultTableModel tableModel;

private JTextField dateField;

private JLabel dateFormatLabel;

private JButton searchButton;

private JButton refreshButton;

private JLabel totalLabel;

private PengunjungDAO pengunjungDAO;

public PengunjungManagementPanel() {

    pengunjungDAO = new PengunjungDAO();

    initComponents();

    setupLayout();

    loadPengunjungData();

    updateTotalCount();

}

private void initComponents() {

    setBackground(BG_GREEN);

}

// Setup table model dengan kolom yang sesuai

String[] columnNames = {"ID Pengunjung", "Nama", "No Telepon", "Tanggal Kunjungan"};

```

```
tableModel = new DefaultTableModel(columnNames, 0) {  
  
    @Override  
  
    public boolean isCellEditable(int row, int column) {  
  
        return false; // Tabel tidak bisa diedit  
  
    }  
  
};  
  
  
// Setup table  
  
pengunjungTable = new JTable(tableModel);  
  
pengunjungTable.setBackground(Color.WHITE);  
  
pengunjungTable.setSelectionBackground(LIGHT_GREEN);  
  
pengunjungTable.setSelectionForeground(DARK_GREEN);  
  
pengunjungTable.setFont(new Font("Segoe UI", Font.PLAIN, 12));  
  
pengunjungTable.setRowHeight(25);  
  
pengunjungTable.setGridColor(PRIMARY_GREEN);  
  
  
  
// Setup table header  
  
JTableHeader header = pengunjungTable.getTableHeader();  
  
header.setBackground(PRIMARY_GREEN);  
  
header.setForeground(Color.WHITE);  
  
header.setFont(new Font("Segoe UI", Font.BOLD, 12));  
  
header.setReorderingAllowed(false);
```

```

// Setup date field untuk pencarian

dateField = new JTextField(15);

dateField.setPreferredSize(new Dimension(150, 35));

dateField.setFont(new Font("Segoe UI", Font.PLAIN, 12));

dateField.setBorder(BorderFactory.createCompoundBorder(
    BorderFactory.createLineBorder(PRIMARY_GREEN, 1),
    BorderFactory.createEmptyBorder(5, 8, 5, 8)
));

// Label untuk format tanggal

dateFormatLabel = new JLabel("(yyyy-mm-dd)");

dateFormatLabel.setFont(new Font("Segoe UI", Font.ITALIC, 10));

dateFormatLabel.setForeground(DARK_GREEN);

// Setup buttons

searchButton = createStyledButton("Cari", PRIMARY_GREEN);

refreshButton = createStyledButton("Refresh", DARK_GREEN);

// Setup label untuk total pengunjung

totalLabel = new JLabel("Total Pengunjung: 0");

totalLabel.setFont(new Font("Segoe UI", Font.BOLD, 14));

```

```
totalLabel.setForeground(DARK_GREEN);

// Setup events

setupEvents();

}

private JButton createStyledButton(String text, Color backgroundColor) {

JButton button = new JButton(text);

button.setBackground(backgroundColor);

button.setForeground(Color.WHITE);

button.setFont(new Font("Segoe UI", Font.BOLD, 12));

button.setPreferredSize(new Dimension(100, 35));

button.setFocusPainted(false);

button.setBorder(BorderFactory.createEmptyBorder(8, 15, 8, 15));

button.setCursor(new Cursor(Cursor.HAND_CURSOR));

button.setOpaque(true);

return button;

}

private void setupLayout() {

setLayout(new BorderLayout(10, 10));

setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
}
```

```

// Panel header dengan judul

JPanel headerPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));

headerPanel.setBackground(BG_GREEN);

JLabel titleLabel = new JLabel("Manajemen Data Pengunjung");

titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 24));

titleLabel.setForeground(DARK_GREEN);

headerPanel.add(titleLabel);

// Panel untuk pencarian

JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 10, 10));

searchPanel.setBackground(BG_GREEN);

searchPanel.setBorder(BorderFactory.createTitledBorder(
    BorderFactory.createLineBorder(PRIMARY_GREEN, 2),
    "Pencarian Berdasarkan Tanggal",
    0, 0, new Font("Segoe UI", Font.BOLD, 12), DARK_GREEN
));

searchPanel.add(new JLabel("Tanggal"));

searchPanel.add(dateField);

searchPanel.add(dateFormatLabel);

```

```

searchPanel.add(searchButton);

searchPanel.add(refreshButton);

// Panel untuk informasi total

JPanel infoPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

infoPanel.setBackground(BG_GREEN);

infoPanel.add(totalLabel);

// Panel bagian atas (header + search + info)

JPanel topPanel = new JPanel(new BorderLayout());

topPanel.setBackground(BG_GREEN);

topPanel.add(headerPanel, BorderLayout.NORTH);

topPanel.add(searchPanel, BorderLayout.CENTER);

topPanel.add(infoPanel, BorderLayout.SOUTH);

// Table dengan scroll pane

JScrollPane scrollPane = new JScrollPane(pengunjungTable);

scrollPane.setBackground(Color.WHITE);

scrollPane.setViewport().setBackground(Color.WHITE);

scrollPane.setBorder(BorderFactory.createLineBorder(PRIMARY_GREEN, 2));

// Layout utama

```

```
add(topPanel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

}

private void setupEvents() {

    searchButton.addActionListener(new ActionListener() {

        @Override

        public void actionPerformed(ActionEvent e) {

            searchByDate();

        }

    });

    refreshButton.addActionListener(new ActionListener() {

        @Override

        public void actionPerformed(ActionEvent e) {

            dateField.setText("");

            loadPengunjungData();

            updateTotalCount();

        }

    });

}
```

```

private void loadPengunjungData() {

    // Clear existing data

    tableModel.setRowCount(0);

    try {

        List<Pengunjung> pengunjungList = pengunjungDAO.getAllPengunjung();

        for (Pengunjung pengunjung : pengunjungList) {

            Object[] row = {

                pengunjung.getIdPengunjung(),

                pengunjung.getNama(),

                pengunjung.getNoTelepon(),

                pengunjung.getTanggalKunjungan()

            };

            tableModel.addRow(row);

        }

    } catch (SQLException e) {

        JOptionPane.showMessageDialog(this,
            "Error saat memuat data pengunjung: " + e.getMessage(),
            "Database Error",
            JOptionPane.ERROR_MESSAGE);
    }
}

```

```

e.printStackTrace();

}

}

private void searchByDate() {

String dateText = dateField.getText().trim();

}

if(dateText.isEmpty()) {

JOptionPane.showMessageDialog(this,
"Silakan masukkan tanggal terlebih dahulu!\nFormat: yyyy-mm-dd (contoh: 2024-01-15)",
"Peringatan",
JOptionPane.WARNING_MESSAGE);

return;
}

// Validasi format tanggal

if(!dateText.matches("\d{4}-\d{2}-\d{2}")) {

JOptionPane.showMessageDialog(this,
"Format tanggal salah!\nGunakan format: yyyy-mm-dd (contoh: 2024-01-15)",
"Format Error",
JOptionPane.ERROR_MESSAGE);

return;
}

```

```

}

// Clear existing data

tableModel.setRowCount(0);

try {

List<Pengunjung> pengunjungList = pengunjungDAO.getPengunjungByDate(dateText);

for (Pengunjung pengunjung : pengunjungList) {

Object[] row = {

pengunjung.getIdPengunjung(),

pengunjung.getNama(),

pengunjung.getNoTelepon(),

pengunjung.getTanggalKunjungan()

};

tableModel.addRow(row);

}

updateTotalCount();

// Show message if no data found

if (tableModel.getRowCount() == 0) {

```

```

JOptionPane.showMessageDialog(this,
    "Tidak ada data pengunjung pada tanggal " + dateText,
    "Informasi",
    JOptionPane.INFORMATION_MESSAGE);
}

} catch (IllegalArgumentException e) {
    JOptionPane.showMessageDialog(this,
        "Format tanggal tidak valid!\nGunakan format: yyyy-mm-dd (contoh: 2024-01-15)",
        "Format Error",
        JOptionPane.ERROR_MESSAGE);
}

} catch (SQLException e) {
    JOptionPane.showMessageDialog(this,
        "Error saat mencari data pengunjung: " + e.getMessage(),
        "Database Error",
        JOptionPane.ERROR_MESSAGE);
    e.printStackTrace();
}
}

private void updateTotalCount() {
    int totalRows = tableView.getRowCount();
}

```

```
totalLabel.setText("Total Pengunjung: " + totalRows);
```

```
}
```

```
}
```

23. Main Method Admin

```
package com.Library.Admin;
```

```
import javax.swing.*;
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        SwingUtilities.invokeLater(() -> {
```

```
            new AdminLoginForm().setVisible(true);
```

```
        });
```

```
    }
```

```
}
```

2.5.2 Anggota

1. Koneksi Database

```
package com.Library.Anggota;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
    public static Connection getConnection() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/db_marimaca_2", "root", ""
            );
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

2. Dashboard Anggota

```
package com.Library.Anggota;

import javax.swing.*;
import javax.swing.border.EmptyBorder;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;

public class DashboardFrame extends JFrame {

    private final Color PRIMARY_GREEN = new Color(67, 160, 71);
    private final Color DARK_GREEN = new Color(46, 125, 50);
    private final Color BG_LIGHT = new Color(248, 252, 248);
    private final Color SIDEBAR_GREEN = new Color(56, 142, 60);
    private final Color HOVER_GREEN = new Color(76, 175, 80);
```

```
private final Color CARD_BG = new Color(255, 255, 255);
private final Color TEXT_PRIMARY = new Color(33, 33, 33);
private final Color TEXT_SECONDARY = new Color(117, 117, 117);
private final Color BORDER_COLOR = new Color(230, 230, 230);
private final Color SUCCESS_GREEN = new Color(56, 142, 60);
private final Color ERROR_RED = new Color(244, 67, 54);

private JPanel mainPanel;
private JPanel contentPanel;
private CardLayout cardLayout;

// Member data variables
private String idAnggotaDisplay;
private String namaAnggota;
private String alamatAnggota;
private String noTeleponAnggota;
private String tanggalDaftarAnggota;
private String emailAnggota;
private String statusAktifAnggota;

private int loggedInMemberId;
```

```
// Sidebar buttons

private JButton homeButton;

private JButton bukuButton;

private JButton riwayatButton;

private JButton profilButton;

public DashboardFrame(int memberId) {

    this.loggedInMemberId = memberId;

    loadMemberData(this.loggedInMemberId);

    initializeFrame();

    setupComponents();

}

private void initializeFrame() {

    setTitle("Dashboard Anggota - Perpustakaan MariMaca");

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setSize(1200, 700);

    setLocationRelativeTo(null);

    setMinimumSize(new Dimension(1000, 600));

    getContentPane().setBackground(BG_LIGHT);

}
```

```

private void setupComponents() {

    mainPanel = new JPanel(new BorderLayout());
    mainPanel.setBackground(BG_LIGHT);
    add(mainPanel);

    JPanel sidebarPanel = createModernSidebarPanel();
    mainPanel.add(sidebarPanel, BorderLayout.WEST);

    cardLayout = new CardLayout();
    contentPanel = new JPanel(cardLayout);
    contentPanel.setBackground(BG_LIGHT);
    contentPanel.setBorder(new EmptyBorder(25, 25, 25, 25));

    JPanel homeContentPanel = createModernHomePanel();
    contentPanel.add(homeContentPanel, "Home");

    BukuViewPanel bukuPanel = new BukuViewPanel();
    contentPanel.add(bukuPanel, "Buku");

    RiwayatPanel riwayatPanel = new RiwayatPanel(loggedInMemberId);
    contentPanel.add(riwayatPanel, "Riwayat");
}

```

```

PengumumanPanel pengumumanPanel = new PengumumanPanel();

contentPanel.add(pengumumanPanel, "Pengumuman");

mainPanel.add(contentPanel, BorderLayout.CENTER);

cardLayout.show(contentPanel, "Home");

}

private void loadMemberData(int memberId) {

    String sql = "SELECT id_anggota, nama, alamat, no_telepon, tanggal_daftar, email,
status_aktif FROM anggota WHERE id_anggota = ?";

    try (Connection conn = DBConnection.getConnection();

        PreparedStatement pstmt = conn.prepareStatement(sql)) {

        pstmt.setInt(1, memberId);

        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {

            this.idAnggotaDisplay = String.valueOf(rs.getInt("id_anggota"));

            this.namaAnggota = rs.getString("nama");

            this.alamatAnggota = rs.getString("alamat");

            this.noTeleponAnggota = rs.getString("no_telepon");

            java.sql.Date dbSqlDate = rs.getDate("tanggal_daftar");

```

```

if (dbSqlDate != null) {

    SimpleDateFormat sdf = new SimpleDateFormat("dd MMMM yyyy");

    this.tanggalDaftarAnggota = sdf.format(dbSqlDate);

} else {

    this.tanggalDaftarAnggota = "Tanggal tidak tersedia";

}

this.emailAnggota = rs.getString("email");

}

int statusInt = rs.getInt("status_aktif");

this.statusAktifAnggota = (statusInt == 1) ? "Aktif" : "Tidak Aktif";

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this,

"Gagal mengambil data anggota: " + e.getMessage(),

"Database Error", JOptionPane.ERROR_MESSAGE);

}

}

private JPanel createModernSidebarPanel() {

JPanel sidebarPanel = new JPanel();

sidebarPanel.setBackground(SIDEBAR_GREEN);

```

```

sidebarPanel.setLayout(new BoxLayout(sidebarPanel, BoxLayout.Y_AXIS));

sidebarPanel.setPreferredSize(new Dimension(260, getHeight()));

sidebarPanel.setBorder(new EmptyBorder(20, 0, 20, 0));

// Logo/Header Section

JPanel headerPanel = createSidebarHeader();

sidebarPanel.add(headerPanel);

sidebarPanel.add(Box.createRigidArea(new Dimension(0, 30)));

// Navigation Buttons

homeButton = createModernSidebarButton("🏠", "Dashboard", true);

homeButton.addActionListener(e -> {

    setActiveButton(homeButton);

    cardLayout.show(contentPanel, "Home");

});

sidebarPanel.add(homeButton);

sidebarPanel.add(Box.createRigidArea(new Dimension(0, 8)));

bukuButton = createModernSidebarButton("📚", "Buku Tersedia", false);

bukuButton.addActionListener(e -> {

    setActiveButton(bukuButton);

    cardLayout.show(contentPanel, "Buku");
});

```

```
});  
  
sidebarPanel.add(bukuButton);  
  
sidebarPanel.add(Box.createRigidArea(new Dimension(0, 8)));  
  
  
riwayatButton = createModernSidebarButton("📖", "Riwayat Pinjaman", false);  
riwayatButton.addActionListener(e -> {  
    setActiveButton(riwayatButton);  
    cardLayout.show(contentPanel, "Riwayat");  
});  
  
sidebarPanel.add(riwayatButton);  
  
sidebarPanel.add(Box.createRigidArea(new Dimension(0, 8)));  
  
  
profilButton = createModernSidebarButton("📢", "Pengumuman", false);  
profilButton.addActionListener(e -> {  
    setActiveButton(profilButton);  
    cardLayout.show(contentPanel, "Pengumuman");  
});  
  
sidebarPanel.add(profilButton);  
  
  
// Spacer  
  
sidebarPanel.add(Box.createVerticalGlue());
```

```

// Logout Button

JButton logoutButton = createLogoutButton();

sidebarPanel.add(logoutButton);

sidebarPanel.add(Box.createRigidArea(new Dimension(0, 20)));

return sidebarPanel;

}

private void setActiveButton(JButton activeButton) {

// Reset all buttons to inactive state

JButton[] buttons = {homeButton, bukuButton, riwayatButton, profilButton};

for (JButton button : buttons) {

button.setBackground(SIDEBAR_GREEN);

}

// Set the clicked button as active

activeButton.setBackground(HOVER_GREEN);

}

private JPanel createSidebarHeader() {

JPanel headerPanel = new JPanel();

headerPanel.setBackground(SIDEBAR_GREEN);

headerPanel.setLayout(new BoxLayout(headerPanel, BoxLayout.Y_AXIS));

```

```
headerPanel.setBorder(new EmptyBorder(0, 20, 0, 20));  
  
JLabel logoLabel = new JLabel("📚");  
logoLabel.setFont(new Font("Segoe UI Emoji", Font.PLAIN, 32));  
logoLabel.setAlignmentX(Component.CENTER_ALIGNMENT);  
headerPanel.add(logoLabel);  
headerPanel.add(Box.createRigidArea(new Dimension(0, 10)));  
  
JLabel titleLabel = new JLabel("MariMaca");  
titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));  
titleLabel.setForeground(Color.WHITE);  
titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);  
headerPanel.add(titleLabel);  
  
JLabel subtitleLabel = new JLabel("Library System");  
subtitleLabel.setFont(new Font("Segoe UI", Font.PLAIN, 12));  
subtitleLabel.setForeground(new Color(200, 230, 201));  
subtitleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);  
headerPanel.add(subtitleLabel);  
  
return headerPanel;  
}
```

```
private JButton createModernSidebarButton(String icon, String text, boolean isActive)
{
    JButton button = new JButton();
    button.setLayout(new BorderLayout());
    button.setBackground(isActive ? HOVER_GREEN : SIDEBAR_GREEN);
    button.setBorder(new EmptyBorder(12, 20, 12, 20));
    button.setFocusPainted(false);
    button.setCursor(new Cursor(Cursor.HAND_CURSOR));

    JLabel iconLabel = new JLabel(icon);
    iconLabel.setFont(new Font("Segoe UI Emoji", Font.PLAIN, 18));
    iconLabel.setForeground(Color.WHITE);
    button.add(iconLabel, BorderLayout.WEST);

    JLabel textLabel = new JLabel(text);
    textLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));
    textLabel.setForeground(Color.WHITE);
    textLabel.setBorder(new EmptyBorder(0, 15, 0, 0));
    button.add(textLabel, BorderLayout.CENTER);

    button.addMouseListener(new MouseAdapter() {
        @Override
```

```

public void mouseEntered(MouseEvent e) {

    if (button.getBackground() != HOVER_GREEN) {

        button.setBackground(HOVER_GREEN);

    }

}

@Override

public void mouseExited(MouseEvent e) {

    // Only reset background if this is not the active button

    if (button != homeButton || homeButton.getBackground() != HOVER_GREEN)

    {

        if (button != bukuButton || bukuButton.getBackground() != HOVER_GREEN) {

            if (button != riwayatButton || riwayatButton.getBackground() != HOVER_GREEN) {

                if (button != profilButton || profilButton.getBackground() != HOVER_GREEN) {

                    button.setBackground(SIDEBAR_GREEN);

                }

            }

        }

    }

}

});


```

```
return button;  
}  
  
private JButton createLogoutButton() {  
    JButton button = new JButton();  
    button.setLayout(new BorderLayout());  
    button.setBackground(new Color(198, 40, 40));  
    button.setBorder(new EmptyBorder(12, 20, 12, 20));  
    button.setFocusPainted(false);  
    button.setCursor(new Cursor(Cursor.HAND_CURSOR));  
  
    JLabel iconLabel = new JLabel(" ");  
    iconLabel.setFont(new Font("Segoe UI Emoji", Font.PLAIN, 18));  
    iconLabel.setForeground(Color.WHITE);  
    button.add(iconLabel, BorderLayout.WEST);  
  
    JLabel textLabel = new JLabel("Keluar");  
    textLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));  
    textLabel.setForeground(Color.WHITE);  
    textLabel.setBorder(new EmptyBorder(0, 15, 0, 0));  
    button.add(textLabel, BorderLayout.CENTER);  
}
```

```
button.addMouseListener(new MouseAdapter() {  
  
    @Override  
  
    public void mouseEntered(MouseEvent e) {  
  
        button.setBackground(new Color(211, 47, 47));  
  
    }  
  
    @Override  
  
    public void mouseExited(MouseEvent e) {  
  
        button.setBackground(new Color(198, 40, 40));  
  
    }  
});  
  
button.addActionListener(e -> {  
  
    int option = JOptionPane.showConfirmDialog(this,  
        "Apakah Anda yakin ingin keluar?",  
        "Konfirmasi Keluar",  
        JOptionPane.YES_NO_OPTION);  
  
    if (option == JOptionPane.YES_OPTION) {  
  
        this.dispose();  
  
    }  
});
```

```

        return button;
    }

private JPanel createModernHomePanel() {
    JPanel panel = new JPanel(new BorderLayout());
    panel.setBackground(BG_LIGHT);

    // Header Section
    JPanel headerPanel = createHeaderSection();
    panel.add(headerPanel, BorderLayout.NORTH);

    // Main Content with Cards
    JPanel mainContentPanel = new JPanel(new GridBagLayout());
    mainContentPanel.setBackground(BG_LIGHT);
    mainContentPanel.setBorder(new EmptyBorder(20, 0, 0, 0));

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(10, 10, 10, 10);
    gbc.fill = GridBagConstraints.BOTH;

    // Profile Card

```

```

gbc.gridx = 0; gbc.gridy = 0;

gbc.weightx = 0.6; gbc.weighty = 1.0;

JPanel profileCard = createProfileCard();

mainContentPanel.add(profileCard, gbc);

// Status Card

gbc.gridx = 1; gbc.gridy = 0;

gbc.weightx = 0.4; gbc.weighty = 1.0;

JPanel statusCard = createStatusCard();

mainContentPanel.add(statusCard, gbc);

panel.add(mainContentPanel, BorderLayout.CENTER);

return panel;
}

private JPanel createHeaderSection() {

JPanel headerPanel = new JPanel(new BorderLayout());

headerPanel.setBackground(BG_LIGHT);

headerPanel.setBorder(new EmptyBorder(0, 0, 20, 0));

JLabel welcomeLabel = new JLabel("Selamat Datang, " + namaAnggota + "!");

```

```

welcomeLabel.setFont(new Font("Segoe UI", Font.BOLD, 28));

welcomeLabel.setForeground(DARK_GREEN);

headerPanel.add(welcomeLabel, BorderLayout.WEST);

JLabel dateLabel = new JLabel(getCurrentDate());

dateLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));

dateLabel.setForeground(TEXT_SECONDARY);

headerPanel.add(dateLabel, BorderLayout.EAST);

return headerPanel;

}

private String getCurrentDate() {

    SimpleDateFormat sdf = new SimpleDateFormat("EEEE, dd MMMM yyyy");

    return sdf.format(new java.util.Date());

}

private JPanel createProfileCard() {

    JPanel card = createCardPanel();

    card.setLayout(new BorderLayout());

    // Card Header

```

```

JPanel headerPanel = new JPanel(new BorderLayout());

headerPanel.setBackground(CARD_BG);

headerPanel.setBorder(new EmptyBorder(0, 0, 20, 0));

JLabel titleLabel = new JLabel("Informasi Profil");

titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));

titleLabel.setForeground(PRIMARY_GREEN);

headerPanel.add(titleLabel, BorderLayout.WEST);

JLabel iconLabel = new JLabel("👤");

iconLabel.setFont(new Font("Segoe UI Emoji", Font.PLAIN, 24));

headerPanel.add(iconLabel, BorderLayout.EAST);

card.add(headerPanel, BorderLayout.NORTH);

// Profile Details

JPanel detailsPanel = new JPanel(new GridBagLayout());

detailsPanel.setBackground(CARD_BG);

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(8, 0, 8, 20);

gbc.anchor = GridBagConstraints.WEST;

```

```
String[] labels = {"ID Anggota", "Nama Lengkap", "Email", "No. Telepon",  
"Alamat", "Tanggal Daftar"};
```

```
String[] values = {idAnggotaDisplay, namaAnggota, emailAnggota,  
noTeleponAnggota, alamatAnggota, tanggalDaftarAnggota};
```

```
for (int i = 0; i < labels.length; i++) {
```

```
    gbc.gridx = 0; gbc.gridy = i;
```

```
    gbc.weightx = 0.3;
```

```
JLabel labelComp = createDetailLabel(labels[i] + ":"");
```

```
detailsPanel.add(labelComp, gbc);
```

```
    gbc.gridx = 1;
```

```
    gbc.weightx = 0.7;
```

```
JLabel valueComp = createDetailValue(values[i]);
```

```
detailsPanel.add(valueComp, gbc);
```

```
}
```

```
card.add(detailsPanel, BorderLayout.CENTER);
```

```
return card;
```

```
}
```

```
private JPanel createStatusCard() {
```

```
JPanel card = createCardPanel();
```

```
card.setLayout(new BorderLayout());  
  
// Card Header  
  
JPanel headerPanel = new JPanel(new BorderLayout());  
  
headerPanel.setBackground(CARD_BG);  
  
headerPanel.setBorder(new EmptyBorder(0, 0, 20, 0));  
  
JLabel titleLabel = new JLabel("Status Keanggotaan");  
  
titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));  
  
titleLabel.setForeground(PRIMARY_GREEN);  
  
headerPanel.add(titleLabel, BorderLayout.WEST);  
  
card.add(headerPanel, BorderLayout.NORTH);  
  
// Status Content  
  
JPanel statusPanel = new JPanel();  
  
statusPanel.setLayout(new BoxLayout(statusPanel, BoxLayout.Y_AXIS));  
  
statusPanel.setBackground(CARD_BG);  
  
// Status Badge  
  
JPanel badgePanel = new JPanel(new FlowLayout(FlowLayout.CENTER));  
  
badgePanel.setBackground(CARD_BG);
```

```

JLabel statusBadge = createStatusBadge();

badgePanel.add(statusBadge);

statusPanel.add(badgePanel);

statusPanel.add(Box.createRigidArea(new Dimension(0, 20)));

// Quick Stats with real data

JPanel statsPanel = new JPanel(new GridLayout(2, 1, 0, 10)); // Changed from 3 to 2
rows

statsPanel.setBackground(CARD_BG);

// Get real data from database

int borrowedBooksCount = getBorrowedBooksCount();

String allDueDates = getAllDueDates();

statsPanel.add(createStatItem("📚", "Buku Dipinjam",
String.valueOf(borrowedBooksCount)));

statsPanel.add(createStatItem("⌚", "Jatuh Tempo", allDueDates));

statusPanel.add(statsPanel);

card.add(statusPanel, BorderLayout.CENTER);

```

```
return card;

}

private JPanel createCardPanel() {

    JPanel card = new JPanel();
    card.setBackground(CARD_BG);
    card.setBorder(BorderFactory.createCompoundBorder(
        BorderFactory.createLineBorder(BORDER_COLOR, 1),
        new EmptyBorder(25, 25, 25, 25)
    ));
    // Add subtle shadow effect
    card.setBorder(BorderFactory.createCompoundBorder(
        BorderFactory.createMatteBorder(0, 0, 3, 3, new Color(0, 0, 0, 20)),
        BorderFactory.createCompoundBorder(
            BorderFactory.createLineBorder(BORDER_COLOR, 1),
            new EmptyBorder(25, 25, 25, 25)
        )
    ));
    return card;
}

private JLabel createDetailLabel(String text) {
```

```

JLabel label = new JLabel(text);

label.setFont(new Font("Segoe UI", Font.BOLD, 14));

label.setForeground(TEXT_PRIMARY);

return label;

}

private JLabel createDetailValue(String text) {

JLabel label = new JLabel(text != null ? text : "Tidak tersedia");

label.setFont(new Font("Segoe UI", Font.PLAIN, 14));

label.setForeground(TEXT_SECONDARY);

return label;

}

private JLabel createStatusBadge() {

JLabel badge = new JLabel("● " + statusAktifAnggota);

badge.setFont(new Font("Segoe UI", Font.BOLD, 16));

badge.setOpaque(true);

badge.setBorder(new EmptyBorder(8, 16, 8, 16));

if ("Aktif".equalsIgnoreCase(statusAktifAnggota)) {

    badge.setBackground(new Color(232, 245, 233));

    badge.setForeground(SUCCESS_GREEN);
}

```

```

} else if ("Tidak Aktif".equalsIgnoreCase(statusAktifAnggota)) {

    badge.setBackground(new Color(255, 235, 238));

    badge.setForeground(ERROR_RED);

} else {

    badge.setBackground(new Color(245, 245, 245));

    badge.setForeground(TEXT_SECONDARY);

}

return badge;
}

private JPanel createStatItem(String icon, String label, String value) {

JPanel panel = new JPanel(new BorderLayout());

panel.setBackground(new Color(248, 250, 252));

panel.setBorder(new EmptyBorder(12, 15, 12, 15));



JLabel iconLabel = new JLabel(icon);

iconLabel.setFont(new Font("Segoe UI Emoji", Font.PLAIN, 16));

panel.add(iconLabel, BorderLayout.WEST);




JPanel textPanel = new JPanel(new BorderLayout());

textPanel.setBackground(new Color(248, 250, 252));

```

```

JLabel labelComp = new JLabel(label);
labelComp.setFont(new Font("Segoe UI", Font.PLAIN, 12));
labelComp.setForeground(TEXT_SECONDARY);
labelComp.setBorder(new EmptyBorder(0, 10, 0, 0));
textPanel.add(labelComp, BorderLayout.NORTH);

JLabel valueComp = new JLabel(value);
valueComp.setFont(new Font("Segoe UI", Font.BOLD, 14));
valueComp.setForeground(TEXT_PRIMARY);
valueComp.setBorder(new EmptyBorder(0, 10, 0, 0));
textPanel.add(valueComp, BorderLayout.SOUTH);

panel.add(textPanel, BorderLayout.CENTER);
return panel;
}

private int getBorrowedBooksCount() {
    String sql = "SELECT COUNT(*) as count FROM peminjaman WHERE
id_anggota = ? AND status_peminjaman = 'Dipinjam'";
    try (Connection conn = DBConnection.getConnection();
        PreparedStatement pstmt = conn.prepareStatement(sql)) {
}

```

```

        pstmt.setInt(1, loggedInMemberId);

        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {
            return rs.getInt("count");
        }

    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this,
                "Gagal mengambil data peminjaman: " + e.getMessage(),
                "Database Error", JOptionPane.ERROR_MESSAGE);
    }

    return 0;
}

private String getAllDueDates() {
    String sql = "SELECT p.tanggal_jatuh_tempo FROM peminjaman p " +
            "JOIN buku b ON p.id_buku = b.id_buku " +
            "WHERE p.id_anggota = ? AND p.status_peminjaman = 'Dipinjam' " +
            "ORDER BY p.tanggal_jatuh_tempo ASC";

    StringBuilder result = new StringBuilder();
    SimpleDateFormat sdf = new SimpleDateFormat("dd MMM yyyy");

```

```
try (Connection conn = DBConnection.getConnection();

      PreparedStatement pstmt = conn.prepareStatement(sql)) {

    pstmt.setInt(1, loggedInMemberId);

    ResultSet rs = pstmt.executeQuery();

    int count = 0;

    while (rs.next()) {

        java.sql.Date dueDate = rs.getDate("tanggal_jatuh_tempo");

        if (dueDate != null) {

            if (count > 0) {

                result.append(", ");

            }

            result.append(sdf.format(dueDate));

            count++;

        }

    }

    if (count == 0) {

        return "Tidak ada";

    }

}
```

```
    }

    return result.toString();

}

} catch (SQLException e) {

    e.printStackTrace();

    JOptionPane.showMessageDialog(this,
        "Gagal mengambil data jatuh tempo: " + e.getMessage(),
        "Database Error", JOptionPane.ERROR_MESSAGE);

}

return "Tidak ada";

}

}
```

3. Book DAO (Data Access Object)

```
package com.Library.Anggota;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import com.Library.Admin.Book;
```

```
public class BookDAO {  
    private Connection connection;  
  
    public BookDAO() {  
        this.connection = DBConnection.getConnection();  
    }  
  
    public List<Book> getAllBooks() {  
        List<Book> books = new ArrayList<>();  
        String sql = "SELECT * FROM buku ORDER BY id_buku";  
  
        try (Statement stmt = connection.createStatement());  
        ResultSet rs = stmt.executeQuery(sql)) {  
  
            while (rs.next()) {  
                Book book = new Book();  
                book.setIdBuku(rs.getInt("id_buku"));  
                book.setJudulBuku(rs.getString("judul_buku"));  
                book.setAuthor(rs.getString("author"));  
                book.setPublisher(rs.getString("publisher"));  
                book.setType(rs.getString("type"));  
                book.setLocation(rs.getString("location"));  
            }  
        }  
    }  
}
```

```
        book.setStatus(rs.getString("status"));

        books.add(book);

    }

} catch (SQLException e) {

    e.printStackTrace();

}

return books;

}

public List<Book> getAvailableBooks() {

    List<Book> books = new ArrayList<>();

    String sql = "SELECT * FROM buku WHERE status = 'Tersedia' ORDER BY
id_buku";

    try (Statement stmt = connection.createStatement()) {

        ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {

                Book book = new Book();

                book.setIdBuku(rs.getInt("id_buku"));

                book.setJudulBuku(rs.getString("judul_buku"));

                book.setAuthor(rs.getString("author"));

            }

        }

    }

}
```

```

        book.setPublisher(rs.getString("publisher"));

        book.setType(rs.getString("type"));

        book.setLocation(rs.getString("location"));

        book.setStatus(rs.getString("status"));

        books.add(book);

    }

} catch (SQLException e) {

    e.printStackTrace();

}

return books;

}

public List<Book> searchBooks(String keyword) {

    List<Book> books = new ArrayList<>();

    String sql = "SELECT * FROM buku WHERE judul_buku LIKE ? OR author LIKE ? OR type LIKE ? OR location LIKE ? ORDER BY id_buku";

    try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

        String searchPattern = "%" + keyword + "%";

        pstmt.setString(1, searchPattern);

        pstmt.setString(2, searchPattern);

        pstmt.setString(3, searchPattern);
    }
}

```

```
pstmt.setString(4, searchPattern);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

    Book book = new Book();

    book.setIdBuku(rs.getInt("id_buku"));

    book.setJudulBuku(rs.getString("judul_buku"));

    book.setAuthor(rs.getString("author"));

    book.setPublisher(rs.getString("publisher"));

    book.setType(rs.getString("type"));

    book.setLocation(rs.getString("location"));

    book.setStatus(rs.getString("status"));

    books.add(book);

}

} catch (SQLException e) {

    e.printStackTrace();

}

return books;

}

public List<Book> getBooksByType(String type) {
```

```
List<Book> books = new ArrayList<>();

String sql = "SELECT * FROM buku WHERE type = ? ORDER BY id_buku";

try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

    pstmt.setString(1, type);

    ResultSet rs = pstmt.executeQuery();

    while (rs.next()) {

        Book book = new Book();

        book.setIdBuku(rs.getInt("id_buku"));

        book.setJudulBuku(rs.getString("judul_buku"));

        book.setAuthor(rs.getString("author"));

        book.setPublisher(rs.getString("publisher"));

        book.setType(rs.getString("type"));

        book.setLocation(rs.getString("location"));

        book.setStatus(rs.getString("status"));

        books.add(book);

    }

} catch (SQLException e) {

    e.printStackTrace();

}
```

```
        return books;  
    }  
  
    public int getTotalBooks() {  
  
        String sql = "SELECT COUNT(*) as total FROM buku";  
  
        try (Statement stmt = connection.createStatement();  
  
             ResultSet rs = stmt.executeQuery(sql)) {  
  
            if (rs.next()) {  
  
                return rs.getInt("total");  
            }  
        } catch (SQLException e) {  
  
            e.printStackTrace();  
        }  
  
        return 0;  
    }  
  
    public int getAvailableBooksCount() {  
  
        String sql = "SELECT COUNT(*) as total FROM buku WHERE status = 'Tersedia';"  
  
        try (Statement stmt = connection.createStatement();  
  
             ResultSet rs = stmt.executeQuery(sql)) {  
  
            if (rs.next()) {  
  
                return rs.getInt("total");  
            }  
        }  
    }  
}
```

```
    }

} catch (SQLException e) {

    e.printStackTrace();

}

return 0;

}

public List<String> getAllTypes() {

    List<String> types = new ArrayList<>();

    String sql = "SELECT DISTINCT type FROM buku WHERE type IS NOT NULL
ORDER BY type";

    try (Statement stmt = connection.createStatement()) {

        ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {

                types.add(rs.getString("type"));

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

    return types;

}
```

```
 }  
 }
```

4. Buku View Panel

```
package com.Library.Anggota;  
  
import javax.swing.*;  
import javax.swing.border.EmptyBorder;  
import javax.swing.table.DefaultTableModel;  
import javax.swing.table.DefaultTableCellRenderer;  
import javax.swing.table.TableRowSorter;  
import java.awt.*;  
import java.util.List;  
import com.Library.Admin.Book;  
  
public class BukuViewPanel extends JPanel {  
  
    private final Color PRIMARY_GREEN = new Color(67, 160, 71);  
    private final Color DARK_GREEN = new Color(46, 125, 50);  
    private final Color LIGHT_GREEN = new Color(129, 199, 132);
```

```
private final Color BG_LIGHT = new Color(248, 252, 248);
private final Color CARD_BG = new Color(255, 255, 255);
private final Color TEXT_PRIMARY = new Color(33, 33, 33);
private final Color TEXT_SECONDARY = new Color(117, 117, 117);
private final Color BORDER_COLOR = new Color(230, 230, 230);
private final Color SUCCESS_GREEN = new Color(56, 142, 60);
private final Color WARNING_ORANGE = new Color(255, 152, 0);

private JTable bukuTable;
private DefaultTableModel tableModel;
private TableRowSorter<DefaultTableModel> sorter;
private JTextField searchField;
private JComboBox<String> filterComboBox;
private JLabel totalBooksLabel;
private JLabel availableBooksLabel;

private BookDAO bookDAO;

public BukuViewPanel() {
    this.bookDAO = new BookDAO();
    initializeComponents();
    loadBooksData();
}
```

```
updateStatistics();

}

private void initializeComponents() {

    setLayout(new BorderLayout());
    setBackground(BG_LIGHT);
    setBorder(new EmptyBorder(25, 25, 25, 25));

}

// Header Panel

JPanel headerPanel = createHeaderPanel();
add(headerPanel, BorderLayout.NORTH);

}

// Filter and Search Panel

JPanel filterPanel = createFilterPanel();
add(filterPanel, BorderLayout.CENTER);

}

// Table Panel

JPanel tablePanel = createTablePanel();
add(tablePanel, BorderLayout.SOUTH);

}

private JPanel createHeaderPanel() {
```

```
JPanel headerPanel = new JPanel(new BorderLayout());  
  
headerPanel.setBackground(BG_LIGHT);  
  
headerPanel.setBorder(new EmptyBorder(0, 0, 20, 0));  
  
  
// Title  
  
JLabel titleLabel = new JLabel("Koleksi Buku Perpustakaan");  
  
titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 24));  
  
titleLabel.setForeground(DARK_GREEN);  
  
headerPanel.add(titleLabel, BorderLayout.WEST);  
  
  
  
// Statistics Panel  
  
JPanel statsPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT, 15, 0));  
  
statsPanel.setBackground(BG_LIGHT);  
  
  
  
totalBooksLabel = createStatLabel("Total Buku: 0", PRIMARY_GREEN);  
  
availableBooksLabel = createStatLabel("Tersedia: 0", SUCCESS_GREEN);  
  
  
  
statsPanel.add(totalBooksLabel);  
  
statsPanel.add(availableBooksLabel);  
  
headerPanel.add(statsPanel, BorderLayout.EAST);  
  
  
  
return headerPanel;
```

```

    }

private JLabel createStatLabel(String text, Color color) {
    JLabel label = new JLabel(text);
    label.setFont(new Font("Segoe UI", Font.BOLD, 14));
    label.setForeground(color);
    label.setOpaque(true);
    label.setBackground(CARD_BG);
    label.setBorder(new EmptyBorder(8, 12, 8, 12));
    return label;
}

private JPanel createFilterPanel() {
    JPanel filterPanel = new JPanel(new BorderLayout());
    filterPanel.setBackground(CARD_BG);
    filterPanel.setBorder(BorderFactory.createCompoundBorder(
        BorderFactory.createLineBorder(BORDER_COLOR, 1),
        new EmptyBorder(20, 20, 20, 20)
    ));
}

// Filter Controls Panel

JPanel controlsPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 15, 0));

```

```
controlsPanel.setBackground(CARD_BG);

// Search Field

JLabel searchLabel = new JLabel("Cari Buku:");
searchLabel.setFont(new Font("Segoe UI", Font.BOLD, 14));
searchLabel.setForeground(TEXT_PRIMARY);
controlsPanel.add(searchLabel);

searchField = new JTextField(20);
searchField.setFont(new Font("Segoe UI", Font.PLAIN, 14));
searchField.setBorder(BorderFactory.createCompoundBorder(
    BorderFactory.createLineBorder(BORDER_COLOR, 1),
    new EmptyBorder(8, 12, 8, 12)
));
controlsPanel.add(searchField);

// Search Button

JButton searchButton = createActionButton("Cari", "");
searchButton.addActionListener(e -> performSearch());
controlsPanel.add(searchButton);

// Filter by Type
```

```

JLabel filterLabel = new JLabel("Filter Type:");
filterLabel.setFont(new Font("Segoe UI", Font.BOLD, 14));
filterLabel.setForeground(TEXT_PRIMARY);
controlsPanel.add(filterLabel);

filterComboBox = new JComboBox<>();
filterComboBox.setFont(new Font("Segoe UI", Font.PLAIN, 14));
filterComboBox.setPreferredSize(new Dimension(150, 35));
loadFilterOptions();
controlsPanel.add(filterComboBox);

// Filter Button
JButton filterButton = createActionButton("Filter", "");
filterButton.addActionListener(e -> performFilter());
controlsPanel.add(filterButton);

// Reset Button
JButton resetButton = createActionButton("Reset", "");
resetButton.addActionListener(e -> resetFilters());
controlsPanel.add(resetButton);

filterPanel.add(controlsPanel, BorderLayout.CENTER);

```

```
return filterPanel;  
}  
  
private JButton createActionButton(String text, String icon) {  
  
    JButton button = new JButton(icon + " " + text);  
  
    button.setFont(new Font("Segoe UI", Font.PLAIN, 12));  
  
    button.setBackground(PRIMARY_GREEN);  
  
    button.setForeground(Color.WHITE);  
  
    button.setFocusPainted(false);  
  
    button.setBorderPainted(false);  
  
    button.setCursor(new Cursor(Cursor.HAND_CURSOR));  
  
    button.setBorder(new EmptyBorder(8, 15, 8, 15));  
  
    button.addMouseListener(new java.awt.event.MouseAdapter() {  
  
        public void mouseEntered(java.awt.event.MouseEvent evt) {  
  
            button.setBackground(DARK_GREEN);  
  
        }  
  
        public void mouseExited(java.awt.event.MouseEvent evt) {  
  
            button.setBackground(PRIMARY_GREEN);  
  
        }  
    });  
}
```

```

        return button;
    }

    private JPanel createTablePanel() {
        JPanel tablePanel = new JPanel(new BorderLayout());
        tablePanel.setBackground(BG_LIGHT);
        tablePanel.setBorder(new EmptyBorder(20, 0, 0, 0));
    }

    // Table Model

    String[] columnNames = {"ID", "Judul Buku", "Author", "Publisher", "Type",
    "Location", "Status"};
    tableModel = new DefaultTableModel(columnNames, 0) {
        @Override
        public boolean isCellEditable(int row, int column) {
            return false;
        }
    };
}

// Table

bukuTable = new JTable(tableModel);
bukuTable.setFont(new Font("Segoe UI", Font.PLAIN, 13));
bukuTable.setRowHeight(35);
bukuTable.setGridColor(BORDER_COLOR);

```

```
bukuTable.setSelectionBackground(LIGHT_GREEN);

bukuTable.setSelectionForeground(TEXT_PRIMARY);

// Table Header

bukuTable.getTableHeader().setFont(new Font("Segoe UI", Font.BOLD, 14));

bukuTable.getTableHeader().setBackground(PRIMARY_GREEN);

bukuTable.getTableHeader().setForeground(Color.WHITE);

bukuTable.getTableHeader().setBorder(BorderFactory.createEmptyBorder());

// Column Widths

bukuTable.getColumnModel().getColumn(0).setPreferredWidth(50); // ID

bukuTable.getColumnModel().getColumn(1).setPreferredWidth(250); // Judul

bukuTable.getColumnModel().getColumn(2).setPreferredWidth(180); // Pengarang

bukuTable.getColumnModel().getColumn(3).setPreferredWidth(150); // Penerbit

bukuTable.getColumnModel().getColumn(4).setPreferredWidth(100); // Tipe

bukuTable.getColumnModel().getColumn(5).setPreferredWidth(120); // Lokasi

bukuTable.getColumnModel().getColumn(6).setPreferredWidth(100); // Status

// Custom renderer for Status column

        bukuTable.getColumnModel().getColumn(6).setCellRenderer(new
StatusCellRenderer());
```

// Table Sorter

```

sorter = new TableRowSorter<>(tableModel);

bukuTable.setRowSorter(sorter);

// Scroll Pane

JScrollPane scrollPane = new JScrollPane(bukuTable);

scrollPane.setBorder(BorderFactory.createLineBorder(BORDER_COLOR, 1));

scrollPane.setViewport().setBackground(CARD_BG);

tablePanel.add(scrollPane, BorderLayout.CENTER);

return tablePanel;

}

private void loadFilterOptions() {

filterComboBox.removeAllItems();

filterComboBox.addItem("Semua Tipe");

List<String> types = bookDAO.getAllTypes();

for (String type : types) {

filterComboBox.addItem(type);

}

}

```

```
private void loadBooksData() {  
    List<Book> allBooks = bookDAO.getAllBooks();  
    loadBooksDataFromList(allBooks);  
}  
  
private void loadBooksDataFromList(List<Book> books) {  
    tableModel.setRowCount(0);  
  
    for (Book book : books) {  
        Object[] row = {  
            book.getIdBuku(),  
            book.getJudulBuku(),  
            book.getAuthor(),  
            book.getPublisher(),  
            book.getType(),  
            book.getLocation(),  
            book.getStatus()  
        };  
        tableModel.addRow(row);  
    }  
}
```

```

private void performSearch() {

    String keyword = searchField.getText().trim();

    if (keyword.isEmpty()) {

        loadBooksData();

    } else {

        List<Book> searchResults = bookDAO.searchBooks(keyword);

        loadBooksDataFromList(searchResults);

    }

    updateStatistics();

}

private void performFilter() {

    String selectedType = (String) filterComboBox.getSelectedItem();

    if ("Semua Tipe".equals(selectedType)) {

        loadBooksData();

    } else {

        List<Book> filteredBooks = bookDAO.getBooksByType(selectedType);

        loadBooksDataFromList(filteredBooks);

    }

    updateStatistics();

}

```

```

private void resetFilters() {
    searchField.setText("");
    filterComboBox.setSelectedIndex(0);
    loadBooksData();
    updateStatistics();
}

private void updateStatistics() {
    int totalBooks = bookDAO.getTotalBooks();
    int availableBooks = bookDAO.getAvailableBooksCount();

    totalBooksLabel.setText("Total Buku: " + totalBooks);
    availableBooksLabel.setText("Tersedia: " + availableBooks);
}

private class StatusCellRenderer extends DefaultTableCellRenderer {
    @Override
    public Component getTableCellRendererComponent(JTable table, Object value,
        boolean isSelected, boolean hasFocus, int row, int column) {
        super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row,
        column);
    }
}

```

```
if (!isSelected) {  
  
    String status = (String) value;  
  
    if ("Tersedia".equalsIgnoreCase(status)) {  
  
        setForeground(SUCCESS_GREEN);  
  
        setFont(getFont().deriveFont(Font.BOLD));  
  
    } else if ("Dipinjam".equalsIgnoreCase(status)) {  
  
        setForeground(WARNING_ORANGE);  
  
        setFont(getFont().deriveFont(Font.BOLD));  
  
    } else {  
  
        setForeground(TEXT_SECONDARY);  
  
        setFont(getFont().deriveFont(Font.PLAIN));  
  
    }  
  
}  
  
return this;
```

5. Login Form

```
package com.Library.Anggota;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class LoginForm extends JFrame {

    private final Color PRIMARY_GREEN = new Color(67, 160, 71);
    private final Color DARK_GREEN = new Color(46, 125, 50);
    private final Color BG_LIGHT = new Color(248, 252, 248);
    private final Color TEXT_COLOR = new Color(50, 50, 50);

    private JTextField usernameField;
    private JPasswordField passwordField;
    private JButton loginButton;
    private JButton registerLinkButton;
```

```
private JLabel errorLabel;

public LoginForm() {
    setTitle("Login Anggota - Perpustakaan MariMaca");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(400, 350);
    setLocationRelativeTo(null);
    getContentPane().setBackground(BG_LIGHT);
    setLayout(new GridBagLayout());
}

initComponents();
addListeners();
}

private void initComponents() {
    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(8, 10, 8, 10);
    gbc.fill = GridBagConstraints.HORIZONTAL;
}

// Title
JLabel titleLabel = new JLabel("LOGIN ANGGOTA", SwingConstants.CENTER);
titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 24));
```

```
titleLabel.setForeground(DARK_GREEN);

gbc.gridx = 0;

gbc.gridy = 0;

gbc.gridwidth = 2;

gbc.insets = new Insets(15, 10, 10, 10);

add(titleLabel, gbc);

gbc.insets = new Insets(8, 10, 8, 10);

// Username Label

JLabel usernameLabel = new JLabel("Username:");

usernameLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));

usernameLabel.setForeground(TEXT_COLOR);

gbc.gridx = 0;

gbc.gridy = 1;

gbc.gridwidth = 1;

add(usernameLabel, gbc);

// Username Field

usernameField = new JTextField(20);

usernameField.setFont(new Font("Segoe UI", Font.PLAIN, 14));

gbc.gridx = 1;
```

```
gbc.gridx = 1;  
add(usernameField, gbc);  
  
// Password Label  
JLabel passwordLabel = new JLabel("Password:");  
passwordLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));  
passwordLabel.setForeground(TEXT_COLOR);  
gbc.gridx = 0;  
gbc.gridy = 2;  
add(passwordLabel, gbc);  
  
// Password Field  
PasswordField = new JPasswordField(20);  
PasswordField.setFont(new Font("Segoe UI", Font.PLAIN, 14));  
gbc.gridx = 1;  
gbc.gridy = 2;  
addPasswordField, gbc);  
  
// Error Label  
errorLabel = new JLabel(" ", SwingConstants.CENTER);  
errorLabel.setFont(new Font("Segoe UI", Font.ITALIC, 12));  
errorLabel.setForeground(Color.RED);
```

```
gbc.gridx = 0;  
  
gbc.gridy = 3;  
  
gbc.gridwidth = 2;  
  
gbc.insets = new Insets(0, 10, 0, 10);  
  
add(errorLabel, gbc);  
  
  
  
gbc.insets = new Insets(8, 10, 8, 10);  
  
  
  
// Login Button  
  
loginButton = new JButton("Login");  
  
loginButton.setFont(new Font("Segoe UI", Font.BOLD, 14));  
  
loginButton.setBackground(PRIMARY_GREEN);  
  
loginButton.setForeground(Color.WHITE);  
  
loginButton.setFocusPainted(false);  
  
loginButton.setPreferredSize(new Dimension(100, 35));  
  
gbc.gridx = 0;  
  
gbc.gridy = 4;  
  
gbc.gridwidth = 2;  
  
gbc.anchor = GridBagConstraints.CENTER;  
  
gbc.fill = GridBagConstraints.NONE;  
  
add(loginButton, gbc);
```

```
// Register Link/Button

registerLinkButton = new JButton("Belum punya akun? Register di sini");

registerLinkButton.setFont(new Font("Segoe UI", Font.PLAIN, 12));

registerLinkButton.setForeground(DARK_GREEN);

registerLinkButton.setBorderPainted(false);

registerLinkButton.setContentAreaFilled(false);

registerLinkButton.setOpaque(false);

registerLinkButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

gbc.gridx = 0;

gbc.gridy = 5;

gbc.gridwidth = 2;

gbc.anchor = GridBagConstraints.CENTER;

gbc.insets = new Insets(0, 10, 10, 10);

add(registerLinkButton, gbc);

}

private void addListeners() {

    loginButton.addActionListener(e -> performLogin());

    passwordField.addActionListener(e -> performLogin());

}

// Listener untuk tombol/link registrasi
```

```
registerLinkButton.addActionListener(new ActionListener() {  
  
    @Override  
  
    public void actionPerformed(ActionEvent e) {  
  
        dispose(); // Tutup form login saat ini  
  
        new RegisterForm().setVisible(true); // Buka form registrasi  
  
    }  
  
});  
  
}  
  
  
  
  
private void performLogin() {  
  
    String username = usernameField.getText().trim();  
  
    String passwordInput = new String(passwordField.getPassword());  
  
  
  
    if (username.isEmpty() || passwordInput.isEmpty()) {  
  
        errorLabel.setText("Username dan password tidak boleh kosong.");  
  
        return;  
  
    }  
  
    errorLabel.setText(" "); // Bersihkan error sebelumnya  
  
  
  
    String sql = "SELECT password, id_anggota FROM user WHERE username = ?";  
  
    try (Connection conn = DBConnection.getConnection();  
  
         PreparedStatement pstmt = conn.prepareStatement(sql)) {
```

```
pstmt.setString(1, username);

ResultSet rs = pstmt.executeQuery();

if (rs.next()) {

    String storedHashedPassword = rs.getString("password");

    int idAnggota = rs.getInt("id_anggota");

}

// Verifikasi password menggunakan PasswordUtil

if (PasswordUtil.verifyPassword(passwordInput, storedHashedPassword)) {

    JOptionPane.showMessageDialog(this,
        "Login berhasil! Selamat datang.",
        "Sukses",
        JOptionPane.INFORMATION_MESSAGE);

}

dispose(); // Tutup LoginForm

}

// Buka DashboardFrame

DashboardFrame dashboard = new DashboardFrame(idAnggota);

dashboard.setVisible(true);

} else {

    errorLabel.setText("Username atau password salah.");
}
```

```

    }

} else {

    errorLabel.setText("Username atau password salah.");

}

}

} catch (SQLException ex) {

    ex.printStackTrace();

    errorLabel.setText("Terjadi kesalahan database. Silakan coba lagi.");

} catch (RuntimeException rex) { // Untuk menangkap error dari PasswordUtil jika ada

    rex.printStackTrace();

    errorLabel.setText("Terjadi kesalahan sistem saat verifikasi.");

}

}

}

```

6. Main

```

package com.Library.Anggota;

import javax.swing.SwingUtilities;

public class Main {

```

```

public static void main(String[] args) {

    SwingUtilities.invokeLater(new Runnable() {

        public void run() {

            LoginForm loginForm = new LoginForm();

            loginForm.setVisible(true);

        }

    });

}

}

```

7. Password Util

```

package com.Library.Anggota;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;

public class PasswordUtil {

    public static String hashPassword(String password) {

        try {

            MessageDigest md = MessageDigest.getInstance("SHA-256");

            byte[] hashedBytes = md.digest(password.getBytes());

```

```

        return Base64.getEncoder().encodeToString(hashedBytes);

    } catch (NoSuchAlgorithmException e) {

        throw new RuntimeException("Gagal melakukan hashing password", e);
    }

}

public static boolean verifyPassword(String inputPassword, String
hashedPasswordFromDB) {

    return hashPassword(inputPassword).equals(hashedPasswordFromDB);
}

}

```

8. Pengumuman Panel

```

package com.Library.Anggota;

import java.awt.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;

```

```
import java.util.Date;  
  
import java.util.List;  
  
import javax.swing.*;  
  
import javax.swing.border.EmptyBorder;  
  
  
public class PengumumanPanel extends JPanel {  
  
  
    private final Color CARD_BG = new Color(255, 255, 255);  
  
    private final Color TEXT_PRIMARY = new Color(33, 33, 33);  
  
    private final Color TEXT_SECONDARY = new Color(117, 117, 117);  
  
    private final Color BORDER_COLOR = new Color(230, 230, 230);  
  
    private final Color BG_LIGHT = new Color(248, 252, 248);  
  
    private final Color PRIMARY_GREEN = new Color(67, 160, 71);  
  
  
    private JPanel announcementsContainer;  
  
  
    private static class Announcement {  
  
        String title;  
  
        String content;  
  
        Date publishDate;  
  
  
        Announcement(String title, String content, Date publishDate, String author) {
```

```
this.title = title;  
this.content = content;  
this.publishDate = publishDate;  
}  
}  
  
public PengumumanPanel() {  
    initializeUI();  
    loadAnnouncements();  
}  
  
private void initializeUI() {  
    setLayout(new BorderLayout());  
    setBackground(BG_LIGHT);  
    setBorder(new EmptyBorder(20, 20, 20, 20));  
  
    JLabel panelTitleLabel = new JLabel("Pengumuman Perpustakaan");  
    panelTitleLabel.setFont(new Font("Segoe UI", Font.BOLD, 24));  
    panelTitleLabel.setForeground(PRIMARY_GREEN);  
    panelTitleLabel.setBorder(new EmptyBorder(0, 0, 20, 0));  
    add(panelTitleLabel, BorderLayout.NORTH);
```

```

announcementsContainer = new JPanel();

    announcementsContainer.setLayout(new BoxLayout(announcementsContainer,
BoxLayout.Y_AXIS));

    announcementsContainer.setBackground(BG_LIGHT);

JScrollPane scrollPane = new JScrollPane(announcementsContainer);

scrollPane.setBorder(BorderFactory.createEmptyBorder());

scrollPane.setBackground(BG_LIGHT);

scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED
);

scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER
);

add(scrollPane, BorderLayout.CENTER);

}

private void loadAnnouncements() {

List<Announcement> announcements = new ArrayList<>();

String sql = "SELECT id_pengumuman, judul, isi, tanggal_dibuat FROM pengumuman
ORDER BY tanggal_dibuat DESC";

try (Connection conn = DBConnection.getConnection()) {

PreparedStatement pstmt = conn.prepareStatement(sql);

ResultSet rs = pstmt.executeQuery() {

```

```

while (rs.next()) {

    String title = rs.getString("judul");

    String content = rs.getString("isi");

    java.sql.Timestamp publishTimestamp = rs.getTimestamp("tanggal_dibuat");

        Date publishDate = (publishTimestamp != null) ? new
Date(publishTimestamp.getTime()) : null;

    announcements.add(new Announcement(title, content, publishDate, "Admin"));

}

}

} catch (SQLException e) {

e.printStackTrace();

// Show error message on the panel itself or a JOptionPane

announcementsContainer.add(new JLabel("Gagal memuat pengumuman: " +
e.getMessage()));

}

}

displayAnnouncements(announcements);

}

private void displayAnnouncements(List<Announcement> announcements) {

announcementsContainer.removeAll(); // Clear previous announcements if any
}

```

```

if (announcements.isEmpty()) {

    JLabel noAnnouncementsLabel = new JLabel("Belum ada pengumuman saat ini.");
    noAnnouncementsLabel.setFont(new Font("Segoe UI", Font.ITALIC, 16));
    noAnnouncementsLabel.setForeground(TEXT_SECONDARY);
    noAnnouncementsLabel.setHorizontalAlignment(SwingConstants.CENTER);

    // Center the label in the available space

    JPanel centerPanel = new JPanel(new GridBagLayout());
    centerPanel.setBackground(BG_LIGHT);
    centerPanel.add(noAnnouncementsLabel);
    announcementsContainer.add(centerPanel);

}

} else {

    for (Announcement announcement : announcements) {

        announcementsContainer.add(createAnnouncementCard(announcement));

        announcementsContainer.add(Box.createRigidArea(new Dimension(0, 15))); // Spacer
    }

    announcementsContainer.revalidate();
    announcementsContainer.repaint();
}

private JPanel createAnnouncementCard(Announcement announcement) {

```

```
JPanel card = new JPanel(new BorderLayout(10, 10));  
card.setBackground(CARD_BG);  
card.setBorder(BorderFactory.createCompoundBorder(  
    BorderFactory.createLineBorder(BORDER_COLOR, 1),  
    new EmptyBorder(15, 20, 15, 20)  
));  
card.setMaximumSize(new Dimension(Integer.MAX_VALUE, 200));  
card.setMinimumSize(new Dimension(300, 100));  
  
JLabel titleLabel = new JLabel(announcement.title);  
titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 18));  
titleLabel.setForeground(TEXT_PRIMARY);  
card.add(titleLabel, BorderLayout.NORTH);  
  
JTextArea contentArea = new JTextArea(announcement.content);  
contentArea.setFont(new Font("Segoe UI", Font.PLAIN, 14));  
contentArea.setForeground(TEXT_SECONDARY);  
contentArea.setWrapStyleWord(true);  
contentArea.setLineWrap(true);  
contentArea.setOpaque(false);  
contentArea.setEditable(false);  
contentArea.setFocusable(false);
```

```

card.add(contentArea, BorderLayout.CENTER);

JPanel footerPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
footerPanel.setBackground(CARD_BG);

SimpleDateFormat sdf = new SimpleDateFormat("dd MMMM yyyy");
String dateStr = (announcement.publishDate != null) ? sdf.format(announcement.publishDate) : "Tanggal tidak diketahui";

JLabel dateLabel = new JLabel("Dipublikasikan : " + dateStr);
dateLabel.setFont(new Font("Segoe UI", Font.ITALIC, 12));
dateLabel.setForeground(TEXT_SECONDARY);
footerPanel.add(dateLabel);

card.add(footerPanel, BorderLayout.SOUTH);

return card;
}
}

```

9. Register Form

```
package com.Library.Anggota;

import javax.swing.*;
import java.awt.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class RegisterForm extends JFrame {

    private final Color PRIMARY_GREEN = new Color(67, 160, 71);
    private final Color DARK_GREEN = new Color(46, 125, 50);
    private final Color BG_LIGHT = new Color(248, 252, 248);
    private final Color TEXT_COLOR = new Color(50, 50, 50);

    private JTextField idAnggotaField;
    private JTextField usernameField;
    private JPasswordField passwordField;
    private JPasswordField confirmPasswordField;
    private JButton registerButton;
```

```
private JButton backToLoginButton;  
  
private JLabel errorLabel;  
  
  
public RegisterForm() {  
  
    setTitle("Registrasi Akun Anggota - Perpustakaan MariMaca");  
  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
    setSize(450, 450);  
  
    setLocationRelativeTo(null);  
  
    getContentPane().setBackground(BG_LIGHT);  
  
    setLayout(new GridBagLayout());  
  
  
    initComponents();  
  
    addListeners();  
  
}  
  
  
private void initComponents() {  
  
    GridBagConstraints gbc = new GridBagConstraints();  
  
    gbc.insets = new Insets(8, 8, 8, 8);  
  
    gbc.fill = GridBagConstraints.HORIZONTAL;  
  
  
    JLabel titleLabel = new JLabel("REGISTRASI AKUN",  
        SwingConstants.CENTER);  
  
    titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 22));
```

```

titleLabel.setForeground(DARK_GREEN);

gbc.gridx = 0; gbc.gridy = 0; gbc.gridwidth = 2;

add(titleLabel, gbc);

// ID Anggota

JLabel idAnggotaLabel = new JLabel("ID Anggota:");

idAnggotaLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));

idAnggotaLabel.setForeground(TEXT_COLOR);

gbc.gridx = 0; gbc.gridy = 1; gbc.gridwidth = 1;

add(idAnggotaLabel, gbc);

idAnggotaField = new JTextField(20);

idAnggotaField.setFont(new Font("Segoe UI", Font.PLAIN, 14));

gbc.gridx = 1; gbc.gridy = 1;

add(idAnggotaField, gbc);

// Username

JLabel usernameLabel = new JLabel("Username Baru:");

usernameLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));

usernameLabel.setForeground(TEXT_COLOR);

gbc.gridx = 0; gbc.gridy = 2;

add(usernameLabel, gbc);

usernameField = new JTextField(20);

```

```
usernameField.setFont(new Font("Segoe UI", Font.PLAIN, 14));
```

```
gbc.gridx = 1; gbc.gridy = 2;
```

```
add(usernameField, gbc);
```

```
// Password
```

```
JLabel passwordLabel = new JLabel("Password Baru:");
```

```
passwordLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));
```

```
passwordLabel.setForeground(TEXT_COLOR);
```

```
gbc.gridx = 0; gbc.gridy = 3;
```

```
add(passwordLabel, gbc);
```

```
passwordField = new JPasswordField(20);
```

```
passwordField.setFont(new Font("Segoe UI", Font.PLAIN, 14));
```

```
gbc.gridx = 1; gbc.gridy = 3;
```

```
add(passwordField, gbc);
```

```
// Konfirmasi Password
```

```
JLabel confirmPasswordLabel = new JLabel("Konfirmasi Password:");
```

```
confirmPasswordLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));
```

```
confirmPasswordLabel.setForeground(TEXT_COLOR);
```

```
gbc.gridx = 0; gbc.gridy = 4;
```

```
add(confirmPasswordLabel, gbc);
```

```
confirmPasswordField = new JPasswordField(20);
```

```

confirmPasswordField.setFont(new Font("Segoe UI", Font.PLAIN, 14));

gbc.gridx = 1; gbc.gridy = 4;

add(confirmPasswordField, gbc);

// Error Label

errorLabel = new JLabel(" ", SwingConstants.CENTER);

errorLabel.setFont(new Font("Segoe UI", Font.ITALIC, 12));

errorLabel.setForeground(Color.RED);

gbc.gridx = 0; gbc.gridy = 5; gbc.gridwidth = 2;

add(errorLabel, gbc);

// Register Button

registerButton = new JButton("Register");

registerButton.setFont(new Font("Segoe UI", Font.BOLD, 14));

registerButton.setBackground(PRIMARY_GREEN);

registerButton.setForeground(Color.WHITE);

gbc.gridx = 0; gbc.gridy = 6; gbc.gridwidth = 2; gbc.anchor = GridBagConstraints.CENTER;

add(registerButton, gbc);

// Back to Login Button

backToLoginButton = new JButton("Kembali ke Login");

backToLoginButton.setFont(new Font("Segoe UI", Font.PLAIN, 12));

```

```

backToLoginButton.setForeground(DARK_GREEN);

backToLoginButton.setBorderPainted(false);

backToLoginButton.setContentAreaFilled(false);

backToLoginButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

        gbc.gridx = 0; gbc.gridy = 7; gbc.gridwidth = 2; gbc.anchor =
GridBagConstraints.CENTER;

        add(backToLoginButton, gbc);

}

private void addListeners() {

    registerButton.addActionListener(e -> performRegistration());

    backToLoginButton.addActionListener(e -> {

        this.dispose();

        new LoginForm().setVisible(true);

    });

}

private void performRegistration() {

    String idAnggotaStr = idAnggotaField.getText().trim();

    String username = usernameField.getText().trim();

    String password = new String(passwordField.getPassword());

    String confirmPassword = new String(confirmPasswordField.getPassword());

```

```

// Validasi Input Kosong

    if (idAnggotaStr.isEmpty() || username.isEmpty() || password.isEmpty() ||
confirmPassword.isEmpty()) {

        errorLabel.setText("Semua field harus diisi!");

        return;
    }

// Validasi Password Match

if (!password.equals(confirmPassword)) {

    errorLabel.setText("Password dan konfirmasi password tidak cocok!");

    return;
}

int idAnggota;

try {

    idAnggota = Integer.parseInt(idAnggotaStr);

} catch (NumberFormatException e) {

    errorLabel.setText("ID Anggota harus berupa angka!");

    return;
}

errorLabel.setText(" "); // Clear error

try (Connection conn = DBConnection.getConnection()) {

```

```
//Cek apakah id_anggota valid / ada di tabel anggota

if (!isIdAnggotaValid(conn, idAnggota)) {

    errorLabel.setText("ID Anggota tidak ditemukan atau tidak valid.");

    return;

}

//Cek apakah id_anggota sudah punya akun di tabel user

if (isIdAnggotaRegistered(conn, idAnggota)) {

    errorLabel.setText("ID Anggota ini sudah memiliki akun terdaftar.");

    return;

}

//Cek apakah username sudah digunakan

if (isUsernameTaken(conn, username)) {

    errorLabel.setText("Username '" + username + "' sudah digunakan. Pilih yang lain.");

    return;

}

// Jika semua validasi lolos, lakukan registrasi

String hashedPassword = PasswordUtil.hashPassword(password);
```

```

String sqlInsert = "INSERT INTO user (id_anggota, username, password)
VALUES (?, ?, ?);"

try (PreparedStatement pstmtInsert = conn.prepareStatement(sqlInsert)) {

    pstmtInsert.setInt(1, idAnggota);

    pstmtInsert.setString(2, username);

    pstmtInsert.setString(3, hashedPassword);

    int rowsAffected = pstmtInsert.executeUpdate();

}

if (rowsAffected > 0) {

    JOptionPane.showMessageDialog(this,
        "Registrasi akun berhasil! Silakan login dengan username dan password
baru Anda.",

        "Registrasi Sukses", JOptionPane.INFORMATION_MESSAGE);

    this.dispose(); // Tutup form registrasi

    new LoginForm().setVisible(true); // Buka form login

} else {

    errorLabel.setText("Registrasi gagal. Silakan coba lagi.");

}

}

}

} catch (SQLException ex) {

    ex.printStackTrace();

    errorLabel.setText("Terjadi kesalahan database saat registrasi.");

```

```

    }

}

private boolean isIdAnggotaValid(Connection conn, int idAnggota) throws
SQLException {

String sql = "SELECT COUNT(*) FROM anggota WHERE id_anggota = ?";

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

    pstmt.setInt(1, idAnggota);

    ResultSet rs = pstmt.executeQuery();

    if (rs.next()) {

        return rs.getInt(1) > 0;

    }

}

return false;
}

private boolean isIdAnggotaRegistered(Connection conn, int idAnggota) throws
SQLException {

String sql = "SELECT COUNT(*) FROM user WHERE id_anggota = ?";

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

    pstmt.setInt(1, idAnggota);

    ResultSet rs = pstmt.executeQuery();

    if (rs.next()) {

```

```

        return rs.getInt(1) > 0;

    }

}

return false;

}

private boolean isUsernameTaken(Connection conn, String username) throws
SQLException {

String sql = "SELECT COUNT(*) FROM user WHERE username = ?";

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

    pstmt.setString(1, username);

    ResultSet rs = pstmt.executeQuery();

    if (rs.next()) {

        return rs.getInt(1) > 0;

    }

}

return false;

}

```

10. Riwayat Pane

```
package com.Library.Anggota;
```

```
import java.awt.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.Vector;
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;

public class RiwayatPanel extends JPanel {

    private final Color PRIMARY_GREEN = new Color(67, 160, 71);
    private final Color BG_LIGHT = new Color(248, 252, 248);
    private final Color TEXT_PRIMARY = new Color(33, 33, 33);
    private final Color TEXT_SECONDARY = new Color(117, 117, 117);
    private final Color BORDER_COLOR = new Color(230, 230, 230);
    private final Color CARD_BG = Color.WHITE;
```

```
private final Color HEADER_BG = new Color(240, 240, 240);

private JTable riwayatTable;
private DefaultTableModel tableModel;
private final int loggedInMemberId;
private JPanel centerContentPanel;
private JLabel emptyMessageLabel;
private JScrollPane scrollPane;

public RiwayatPanel(int memberId) {
    this.loggedInMemberId = memberId;
    initComponents();
    loadRiwayatPeminjaman();
}

private void initComponents() {
    setBackground(BG_LIGHT);
    setLayout(new BorderLayout(15, 15));
    setBorder(new EmptyBorder(25, 25, 25, 25));
}

JLabel titleLabel = new JLabel("Riwayat Peminjaman Anda");
titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 24));
```

```

titleLabel.setForeground(PRIMARY_GREEN);

titleLabel.setHorizontalTextPosition(SwingConstants.LEFT);

add(titleLabel, BorderLayout.NORTH);

// Setup Table

String[] columnNames = {"Judul Buku", "Tgl Pinjam", "Jatuh Tempo", "Tgl
Kembali", "Status"};

tableModel = new DefaultTableModel(columnNames, 0) {

    @Override

    public boolean isCellEditable(int row, int column) {

        return false; // Membuat sel tabel tidak bisa diedit

    }

};

riwayatTable = new JTable(tableModel);

setupTableStyle();

scrollPane = new JScrollPane(riwayatTable);

scrollPane.setBorder(BorderFactory.createLineBorder(BORDER_COLOR));

scrollPane.setViewport().setBackground(CARD_BG);

// Label untuk pesan jika tidak ada riwayat

emptyMessageLabel = new JLabel("Anda belum memiliki riwayat peminjaman.",
SwingConstants.CENTER);

```

```

emptyMessageLabel.setFont(new Font("Segoe UI", Font.ITALIC, 16));

emptyMessageLabel.setForeground(TEXT_SECONDARY);

// Panel tengah untuk menampung tabel atau pesan kosong

centerContentPanel = new JPanel(new CardLayout());

centerContentPanel.setBackground(BG_LIGHT);

centerContentPanel.add(scrollPane, "TABLE_VIEW");

centerContentPanel.add(emptyMessageLabel, "EMPTY_VIEW");

add(centerContentPanel, BorderLayout.CENTER);

}

private void setupTableStyle() {

    riwayatTable.setFont(new Font("Segoe UI", Font.PLAIN, 13));

    riwayatTable.setRowHeight(30);

    riwayatTable.setGridColor(BORDER_COLOR);

    riwayatTable.setShowGrid(true);

    riwayatTable.setFillsViewportHeight(true); // Tabel mengisi tinggi scroll pane

    riwayatTable.setSelectionBackground(PRIMARY_GREEN.brighter());

    riwayatTable.setSelectionForeground(Color.WHITE);

    riwayatTable.setIntercellSpacing(new Dimension(0, 1)); // Menghilangkan spasi
horizontal antar sel
}

```

```

JTableHeader header = riwayatTable.getTableHeader();

header.setFont(new Font("Segoe UI", Font.BOLD, 14));

header.setBackground(HIGHLIGHT_BGCOLOR);
header.setForeground(TEXT_PRIMARY);

header.setBorder(BorderFactory.createLineBorder(BORDER_COLOR));

header.setReorderingAllowed(false);

// Atur lebar kolom

TableColumn column;

column = riwayatTable.getColumnModel().getColumn(0); // Judul Buku

column.setPreferredWidth(250);

column = riwayatTable.getColumnModel().getColumn(1); // Tgl Pinjam

column.setPreferredWidth(100);

column = riwayatTable.getColumnModel().getColumn(2); // Jatuh Tempo

column.setPreferredWidth(100);

column = riwayatTable.getColumnModel().getColumn(3); // Tgl Kembali

column.setPreferredWidth(100);

column = riwayatTable.getColumnModel().getColumn(4); // Status

column.setPreferredWidth(100);

}

public void loadRiwayatPeminjaman() {

```

```

tableModel.setRowCount(0); // Bersihkan data tabel sebelum memuat yang baru

String sql = "SELECT b.judul_buku, p.tanggal_pinjam, p.tanggal_jatuh_tempo, " +
    "p.tanggal_pengembalian, p.status_peminjaman " +
    "FROM peminjaman p " +
    "JOIN buku b ON p.id_buku = b.id_buku " +
    "WHERE p.id_anggota = ? " +
    "ORDER BY p.tanggal_pinjam DESC";

```



```

SimpleDateFormat sdf = new SimpleDateFormat("dd MMM yyyy"); // Format
tanggal

boolean hasData = false;

```



```

try (Connection conn = DBConnection.getConnection()) {

    PreparedStatement pstmt = conn.prepareStatement(sql)) {

        pstmt.setInt(1, loggedInMemberId);

        ResultSet rs = pstmt.executeQuery();

        while (rs.next()) {

            hasData = true;

            Vector<String> row = new Vector<>();

            row.add(rs.getString("judul_buku"));

```

```

        java.sql.Date tglPinjam = rs.getDate("tanggal_pinjam");

        row.add(tglPinjam != null ? sdf.format(tglPinjam) : "N/A");

        java.sql.Date tglJatuhTempo = rs.getDate("tanggal_jatuh_tempo");

        row.add(tglJatuhTempo != null ? sdf.format(tglJatuhTempo) : "N/A");

        java.sql.Date tglKembali = rs.getDate("tanggal_pengembalian");

        row.add(tglKembali != null ? sdf.format(tglKembali) : "Belum Kembali");

        row.add(rs.getString("status_peminjaman"));

        tableModel.addRow(row);

    }

} catch (SQLException e) {

    e.printStackTrace();

    JOptionPane.showMessageDialog(this,
        "Gagal mengambil data riwayat peminjaman: " + e.getMessage(),
        "Database Error", JOptionPane.ERROR_MESSAGE);

    hasData = false; // Jika error, anggap tidak ada data untuk UI

}

```

```

CardLayout cl = (CardLayout) centerContentPanel.getLayout();

if (hasData) {

    cl.show(centerContentPanel, "TABLE_VIEW");

} else {

    // Jika tidak ada data atau terjadi error, pastikan tabel kosong dan tampilkan pesan

    if (tableModel.getRowCount() > 0) tableModel.setRowCount(0);

    cl.show(centerContentPanel, "EMPTY_VIEW");

}

}

// Metode untuk memuat ulang data

public void reloadData() {

    loadRiwayatPeminjaman();

}

}

```

2.5.3 Welcome

1. Library Dashboard

```

package com.Library.Welcome;

import javax.swing.*;
import java.awt.*;

```

```
import java.awt.event.*;
import java.sql.*;
public class LibraryDashboard extends JFrame {
    private final Color PRIMARY_GREEN = new Color(67, 160, 71);
    private final Color DARK_GREEN = new Color(46, 125, 50);
    private final Color SOFT_GREEN = new Color(129, 199, 132);
    private final Color BG_LIGHT = new Color(248, 252, 248);
    private final Color HOVER_GREEN = new Color(76, 175, 80);
    private JTextField searchField;
    private JPanel pengumumanPanel;
    private JPanel bukuPanel;
    private JTabbedPane tabbedPane;
    public LibraryDashboard() {
        setTitle("Dashboard Perpustakaan");
        setExtendedState(JFrame.MAXIMIZED_BOTH);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel mainPanel = new JPanel(new BorderLayout());
        mainPanel.setBackground(BG_LIGHT);
```

```

JPanel topPanel = new JPanel(new BorderLayout());
topPanel.setBackground(DARK_GREEN);
topPanel.setPreferredSize(new Dimension(0, 80));
topPanel.setBorder(BorderFactory.createEmptyBorder(10, 30, 10, 30));

JLabel logoLabel = new JLabel("MARIMACA");
logoLabel.setFont(new Font("SansSerif", Font.BOLD, 26));
logoLabel.setForeground(Color.WHITE);
ImageIcon logoIcon = new ImageIcon("logo.jpg");
if (logoIcon.getImageLoadStatus() == MediaTracker.COMPLETE) {
    logoLabel.setIcon(new ImageIcon(logoIcon.getImage().getScaledInstance(50, 50,
Image.SCALE_SMOOTH)));
    logoLabel.setIconTextGap(15);
}
topPanel.add(logoLabel, BorderLayout.WEST);

JPanel rightTopPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT, 15, 10));
rightTopPanel.setBackground(DARK_GREEN);

searchField = new JTextField(30);
searchField.setFont(new Font("SansSerif", Font.PLAIN, 14));
JButton searchButton = new JButton("Search");

```

```
searchButton.setBackground(PRIMARY_GREEN);

searchButton.setForeground(Color.WHITE);

searchButton.setFont(new Font("SansSerif", Font.BOLD, 14));

searchButton.setFocusPainted(false);

searchButton.addMouseListener(new MouseAdapter() {

    public void mouseEntered(MouseEvent evt) {

        searchButton.setBackground(HOVER_GREEN);

    }

    public void mouseExited(MouseEvent evt) {

        searchButton.setBackground(PRIMARY_GREEN);

    }

});;

JButton backButton = new JButton("Kembali");

backButton.setFont(new Font("SansSerif", Font.BOLD, 14));

backButton.setBackground(Color.RED);

backButton.setForeground(Color.WHITE);

backButton.setFocusPainted(false);

backButton.addActionListener(e -> {

    new Welcome().setVisible(true);

    this.dispose();

});
```

```
});  
  
rightTopPanel.add(searchField);  
rightTopPanel.add(searchButton);  
rightTopPanel.add(backButton);  
topPanel.add(rightTopPanel, BorderLayout.EAST);  
  
JLabel titleLabel = new JLabel("Dashboard Perpustakaan");  
titleLabel.setFont(new Font("SansSerif", Font.BOLD, 28));  
titleLabel.setBorder(BorderFactory.createEmptyBorder(20, 30, 20, 10));  
  
tabbedPane = new JTabbedPane();  
tabbedPane.setFont(new Font("SansSerif", Font.BOLD, 16));  
  
pengumumanPanel = new JPanel(new GridLayout(0, 2, 20, 20));  
pengumumanPanel.setBackground(BG_LIGHT);  
pengumumanPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));  
  
JScrollPane pengumumanScroll = new JScrollPane(pengumumanPanel);  
pengumumanScroll.setBorder(null);  
pengumumanScroll.getVerticalScrollBar().setUnitIncrement(16);  
tabbedPane.addTab("Pengumuman", pengumumanScroll);
```

```
// --- Changes for bukuPanel ---  
  
bukuPanel = new JPanel(new GridLayout(0, 4, 15, 15)); // Changed to 4 columns,  
reduced gaps  
  
bukuPanel.setBackground(BG_LIGHT);  
  
bukuPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));  
  
  
JScrollPane bukuScroll = new JScrollPane(bukuPanel);  
  
bukuScroll.setBorder(null);  
  
bukuScroll.getVerticalScrollBar().setUnitIncrement(16);  
  
tabbedPane.addTab("Daftar Pustaka", bukuScroll);  
  
  
  
searchButton.addActionListener(e -> {  
  
    String keyword = searchField.getText();  
  
    if (tabbedPane.getSelectedIndex() == 0) {  
  
        loadPengumuman(keyword);  
  
    } else {  
  
        loadBuku(keyword);  
  
    }  
  
});  
  
  
tabbedPane.addChangeListener(e -> {  
  
    String keyword = searchField.getText();
```

```
if (tabbedPane.getSelectedIndex() == 0) {  
    loadPengumuman(keyword);  
} else {  
    loadBuku(keyword);  
}  
});  
  
JPanel topWrapper = new JPanel(new BorderLayout());  
topWrapper.setBackground(BG_LIGHT);  
topWrapper.add(topPanel, BorderLayout.NORTH);  
topWrapper.add(titleLabel, BorderLayout.SOUTH);  
  
mainPanel.add(topWrapper, BorderLayout.NORTH);  
mainPanel.add(tabbedPane, BorderLayout.CENTER);  
  
getContentPane().add(mainPanel);  
loadPengumuman("");  
loadBuku("");  
}  
  
private void loadPengumuman(String keyword) {  
    pengumumanPanel.removeAll();
```

```

        try      (Connection      conn      =
DriverManager.getConnection("jdbc:mysql://localhost:3306/db_marimaca_2", "root", ""))
{

    String query = "SELECT * FROM pengumuman WHERE judul LIKE ?";

    PreparedStatement stmt = conn.prepareStatement(query);

    stmt.setString(1, "%" + keyword + "%");

    ResultSet rs = stmt.executeQuery();

    while (rs.next()) {

        JPanel card = createCard(
            rs.getString("judul"),
            rs.getString("isi"),
            rs.getDate("tanggal_dibuat").toString()
        );

        pengumumanPanel.add(card);
    }

    rs.close();
    stmt.close();
} catch (SQLException e) {

    JOptionPane.showMessageDialog(this, "Gagal memuat pengumuman: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
}

```

```
pengumumanPanel.revalidate();
pengumumanPanel.repaint();
}

private void loadBuku(String keyword) {
    bukuPanel.removeAll();

    try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/db_marimaca_2", "root", ""));
    {
        String query = "SELECT * FROM buku WHERE judul_buku LIKE ? OR author LIKE ? OR publisher LIKE ? OR type LIKE ? OR location LIKE ? OR status LIKE ?";
        PreparedStatement stmt = conn.prepareStatement(query);
        for (int i = 1; i <= 6; i++) stmt.setString(i, "%" + keyword + "%");

        ResultSet rs = stmt.executeQuery();
        while (rs.next()) {
            JPanel card = createBookCard(
                rs.getString("judul_buku"),
                rs.getString("author"),
                rs.getString("publisher"),
                rs.getString("type"),
                rs.getString("location"),
                rs.getString("status")
            );
            bukuPanel.add(card);
        }
    }
}
```

```

        rs.getString("location"),
        rs.getString("status")
    );
    bukuPanel.add(card);
}

rs.close();
stmt.close();
} catch (SQLException e) {
    JOptionPane.showMessageDialog(this, "Gagal memuat daftar buku: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
}

bukuPanel.revalidate();
bukuPanel.repaint();
}

private JPanel createCard(String title, String content, String date) {
    JPanel card = new JPanel();
    card.setLayout(new BoxLayout(card, BoxLayout.Y_AXIS));
    card.setBackground(Color.WHITE);
    card.setBorder(BorderFactory.createCompoundBorder(
        BorderFactory.createLineBorder(SOFT_GREEN, 1),

```

```
BorderFactory.createEmptyBorder(15, 15, 15, 15)));  
card.setMaximumSize(new Dimension(400, 200));  
  
JLabel titleLabel = new JLabel(title);  
titleLabel.setFont(new Font("SansSerif", Font.BOLD, 18));  
titleLabel.setAlignmentX(Component.LEFT_ALIGNMENT);  
  
JTextArea contentArea = new JTextArea(content);  
contentArea.setLineWrap(true);  
contentArea.setWrapStyleWord(true);  
contentArea.setEditable(false);  
contentArea.setOpaque(false);  
contentArea.setFont(new Font("SansSerif", Font.PLAIN, 14));  
contentArea.setAlignmentX(Component.LEFT_ALIGNMENT);  
  
JLabel dateLabel = new JLabel(date);  
dateLabel.setFont(new Font("SansSerif", Font.ITALIC, 12));  
dateLabel.setAlignmentX(Component.LEFT_ALIGNMENT);  
dateLabel.setForeground(Color.GRAY);  
  
card.add(Box.createVerticalStrut(5));  
card.add(titleLabel);
```

```

card.add(Box.createVerticalStrut(10));

card.add(contentArea);

card.add(Box.createVerticalGlue());

card.add(dateLabel);

return card;

}

private JPanel createBookCard(String title, String author, String publisher, String type,
String location, String status) {

JPanel card = new JPanel();

card.setLayout(new BoxLayout(card, BoxLayout.Y_AXIS));

card.setBackground(Color.WHITE);

card.setBorder(BorderFactory.createCompoundBorder(
    BorderFactory.createLineBorder(SOFT_GREEN, 1),
    BorderFactory.createEmptyBorder(10, 10, 10, 10))); // Reduced padding

card.setMaximumSize(new Dimension(250, 180)); // Smaller dimensions

card.setPreferredSize(new Dimension(250, 180)); // Set preferred size as well for
consistent layout

JLabel titleLabel = new JLabel(title);

titleLabel.setFont(new Font("SansSerif", Font.BOLD, 14)); // Smaller font

titleLabel.setAlignmentX(Component.LEFT_ALIGNMENT);

```

```

JLabel authorLabel = new JLabel("Penulis: " + author);
JLabel publisherLabel = new JLabel("Penerbit: " + publisher);
JLabel typeLabel = new JLabel("Jenis: " + type);
JLabel locationLabel = new JLabel("Lokasi: " + location);
JLabel statusLabel = new JLabel("Status: " + status);

for (JLabel label : new JLabel[]{authorLabel, publisherLabel, typeLabel,
locationLabel, statusLabel}) {
    label.setFont(new Font("SansSerif", Font.PLAIN, 12)); // Smaller font
    label.setAlignmentX(Component.LEFT_ALIGNMENT);
}

statusLabel.setFont(new Font("SansSerif", Font.BOLD, 12)); // Smaller font
if (status.equalsIgnoreCase("Tersedia")) {
    statusLabel.setForeground(new Color(0, 100, 0));
} else if (status.equalsIgnoreCase("Dipinjam")) {
    statusLabel.setForeground(new Color(139, 0, 0));
} else {
    statusLabel.setForeground(Color.BLACK);
}

card.add(Box.createVerticalStrut(3)); // Reduced strut

```

```

card.add(titleLabel);

card.add(Box.createVerticalStrut(7)); // Reduced strut

card.add(authorLabel);

card.add(publisherLabel);

card.add(typeLabel);

card.add(locationLabel);

card.add(Box.createVerticalGlue());

card.add(statusLabel);

return card;

}

public static void main(String[] args) {

    SwingUtilities.invokeLater(() -> {

        LibraryDashboard dashboard = new LibraryDashboard();

        dashboard.setVisible(true);

        dashboard.setLocationRelativeTo(null);

    });

}

}

```

2. Welcome

```
package com.Library.Welcome;

import javax.swing.*;
import java.awt.*;
import java.sql.*;

public class Welcome extends JFrame {

    private JTextField namaField;
    private JTextField nomorField;

    public Welcome() {
        setTitle("Kunjungan Perpustakaan");
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        ImageIcon bgImage = new ImageIcon(getClass().getResource("background.png"));
        ImageIcon logoIcon = new ImageIcon(getClass().getResource("logo.jpg"));

        JPanel backgroundPanel = new JPanel() {
            protected void paintComponent(Graphics g) {
```

```
super.paintComponent(g);

g.drawImage(bgImage.getImage(), 0, 0, getWidth(), getHeight(), this);

};

};

backgroundPanel.setLayout(new BoxLayout(backgroundPanel,
BoxLayout.Y_AXIS));

backgroundPanel.setBorder(BorderFactory.createEmptyBorder(40, 60, 40, 60));

JLabel title = new JLabel("SELAMAT DATANG", SwingConstants.CENTER);

title.setForeground(Color.WHITE);

title.setFont(new Font("SansSerif", Font.BOLD, 24));

title.setAlignmentX(Component.CENTER_ALIGNMENT);

JLabel subtitle = new JLabel("DI PERPUSTAKAAN", SwingConstants.CENTER);

subtitle.setForeground(Color.WHITE);

subtitle.setFont(new Font("SansSerif", Font.BOLD, 20));

subtitle.setAlignmentX(Component.CENTER_ALIGNMENT);

JLabel instruksi = new JLabel("Jika Anda ingin mendaftarkan diri sebagai anggota Perpustakaan silahkan tanyakan ke petugas.");

instruksi.setForeground(Color.WHITE);

instruksi.setFont(new Font("SansSerif", Font.PLAIN, 14));
```

```
instruksi.setAlignmentX(Component.CENTER_ALIGNMENT);

JLabel instruksi2 = new JLabel("Silakan masukkan nama lengkap dan nomor telepon Anda.");
instruksi2.setForeground(Color.WHITE);
instruksi2.setFont(new Font("SansSerif", Font.PLAIN, 14));
instruksi2.setAlignmentX(Component.CENTER_ALIGNMENT);

JLabel website = new JLabel("DATA KUNJUNGAN HARIAN",
SwingConstants.CENTER);
website.setForeground(Color.WHITE);
website.setFont(new Font("SansSerif", Font.BOLD, 16));
website.setAlignmentX(Component.CENTER_ALIGNMENT);

JLabel logo = new JLabel(new ImageIcon(logoIcon.getImage().getScaledInstance(200,
200,
Image.SCALE_SMOOTH)));
logo.setAlignmentX(Component.CENTER_ALIGNMENT);

JPanel inputPanel = new JPanel(new GridBagLayout());
inputPanel.setOpaque(false);

GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(10, 20, 10, 20);
```

```
gbc.gridx = 0;

JPanel namaPanel = new JPanel();
namaPanel.setLayout(new BoxLayout(namaPanel, BoxLayout.Y_AXIS));
namaPanel.setOpaque(false);
namaPanel.setAlignmentX(Component.CENTER_ALIGNMENT);

JLabel namaLabel = new JLabel("Nama Lengkap", SwingConstants.CENTER);
namaLabel.setForeground(Color.WHITE);
namaLabel.setFont(new Font("SansSerif", Font.BOLD, 14));
namaLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

namaField = createRoundedTextField("");
namaField.setMaximumSize(new Dimension(200, 40));
namaField.setAlignmentX(Component.CENTER_ALIGNMENT);

namaPanel.add(namaLabel);
namaPanel.add(Box.createVerticalStrut(5));
namaPanel.add(namaField);

gbc.gridx = 0;
inputPanel.add(namaPanel, gbc);
```

```
JPanel nomorPanel = new JPanel();
nomorPanel.setLayout(new BoxLayout(nomorPanel, BoxLayout.Y_AXIS));
nomorPanel.setOpaque(false);
nomorPanel.setAlignmentX(Component.CENTER_ALIGNMENT);

JLabel nomorLabel = new JLabel("Nomor Telepon", SwingConstants.CENTER);
nomorLabel.setForeground(Color.WHITE);
nomorLabel.setFont(new Font("SansSerif", Font.BOLD, 14));
nomorLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

nomorField = createRoundedTextField("");
nomorField.setMaximumSize(new Dimension(200, 40));
nomorField.setAlignmentX(Component.CENTER_ALIGNMENT);

nomorPanel.add(nomorLabel);
nomorPanel.add(Box.createVerticalStrut(5));
nomorPanel.add(nomorField);

gbc.gridx = 1;
inputPanel.add(nomorPanel, gbc);
inputPanel.setAlignmentX(Component.CENTER_ALIGNMENT);
```

```
inputPanel.setMaximumSize(new Dimension(600, 100));  
  
JButton submitButton = new JButton("TAMBAH") {  
  
    protected void paintComponent(Graphics g) {  
  
        if (getModel().isPressed()) {  
  
            g.setColor(new Color(30, 30, 30));  
  
        } else {  
  
            g.setColor(new Color(50, 50, 50));  
  
        }  
  
        g.fillRoundRect(0, 0, getWidth(), getHeight(), 40, 40);  
  
        super.paintComponent(g);  
  
    }  
  
    protected void paintBorder(Graphics g) {  
  
        g.setColor(Color.DARK_GRAY);  
  
        g.drawRoundRect(0, 0, getWidth()-1, getHeight()-1, 40, 40);  
  
    }  
};  
  
submitButton.setForeground(Color.WHITE);  
  
submitButton.setFont(new Font("SansSerif", Font.BOLD, 14));  
  
submitButton.setFocusPainted(false);  
  
submitButton.setContentAreaFilled(false);
```

```

submitButton.setOpaque(false);

submitButton.setPreferredSize(new Dimension(150, 40));

submitButton.setMaximumSize(new Dimension(150, 40));

submitButton.setAlignmentX(Component.CENTER_ALIGNMENT);

submitButton.addActionListener(e -> {

    String nama = namaField.getText();

    String nomor = nomorField.getText();

    if (nama.isEmpty() || nomor.isEmpty()) {

        JOptionPane.showMessageDialog(null, "Field tidak boleh kosong");

    } else {

        try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/db_marimaca_1", "root", ""))

    {

        String query = "INSERT INTO pengunjung (nama, no_telepon) VALUES (?, ?)";

        PreparedStatement stmt = conn.prepareStatement(query);

        stmt.setString(1, nama);

        stmt.setString(2, nomor);

        stmt.executeUpdate();

    }

    stmt.close();
}

```

```
        dispose();

        new LibraryDashboard().setVisible(true);

    } catch (SQLException ex) {

        JOptionPane.showMessageDialog(null, "Gagal menyimpan ke database: " +
ex.getMessage());

    }

}

});

backgroundPanel.add(title);

backgroundPanel.add(Box.createRigidArea(new Dimension(0, 5)));

backgroundPanel.add(subtitle);

backgroundPanel.add(Box.createRigidArea(new Dimension(0, 10)));

backgroundPanel.add(instruksi);

backgroundPanel.add(Box.createRigidArea(new Dimension(0, 5)));

backgroundPanel.add(instruksi2);

backgroundPanel.add(Box.createRigidArea(new Dimension(0, 5)));

backgroundPanel.add(website);

backgroundPanel.add(Box.createRigidArea(new Dimension(0, 20)));

backgroundPanel.add(logo);

backgroundPanel.add(Box.createRigidArea(new Dimension(0, 20)));

backgroundPanel.add(inputPanel);

backgroundPanel.add(Box.createRigidArea(new Dimension(0, 20))));
```

```
backgroundPanel.add(submitButton);

setContentPane(backgroundPanel);

}

private JTextField createRoundedTextField(String hint) {
    JTextField field = new JTextField() {
        protected void paintComponent(Graphics g) {
            g.setColor(new Color(255, 255, 255, 230));
            g.fillRoundRect(0, 0, getWidth()-1, getHeight()-1, 40, 40);
            super.paintComponent(g);
        }
    };
    protected void paintBorder(Graphics g) {
        g.setColor(Color.LIGHT_GRAY);
        g.drawRoundRect(0, 0, getWidth()-1, getHeight()-1, 40, 40);
    }
};

field.setFont(new Font("SansSerif", Font.PLAIN, 14));
field.setHorizontalAlignment(SwingConstants.CENTER);
field.setPreferredSize(new Dimension(200, 40));
field.setMaximumSize(new Dimension(200, 40));
```

```
    field.setText("");
    field.setToolTipText(hint);
    field.setOpaque(false);
    return field;
}
}
```

3. Main

```
package com.Library.Welcome;

import javax.swing.*;

public class Main {
    public static void main(String[] args) {
        // try {
        //     UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        // } catch (Exception e) {
        //     e.printStackTrace();
        // }
        // Jalankan aplikasi di Event Dispatch Thread
    }
}
```

```
SwingUtilities.invokeLater(() -> {  
    new Welcome().setVisible(true);  
});  
}  
}
```

2.6 Graphical User Interface (GUI)

2.6.1 Tampilan Admin



Gambar 3 Tampilan home

Dashboard Admin - Sistem Perpustakaan Mari Maca

Admin

Daftar Buku						
ID	Judul Buku	Author	Publisher	Type	Location	Status
1	Laskar Pelangi	Andrea Hirata	Bentang Pustaka	Novel	A1	Dipinjam
2	Bumi Manusia	Pramoedya Ananta Toer	Lentera Dipantara	Novel	A1	Dipinjam
3	Negeri 5 Menara	A. Fuadi	Republika	Novel	A1	Dipinjam
4	Gadis Kretek	Ratih Kumala	Gramedia	Novel	A1	Tersedia
5	Sang Pemimpin	Andrea Hirata	Bentang Pustaka	Novel	A1	Tersedia
6	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Bloomsbury	Novel	A1	Tersedia
7	The Hobbit	J.R.R. Tolkien	Allen & Unwin	Novel	A1	Tersedia
8	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	Novel	A1	Tersedia
9	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	Novel	A1	Tersedia
10	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	Novel	A1	Rusak
11	Ensiklopedia Dunia	Various	Gramedia	Ensiklopedia	A2	Tersedia
12	Ensiklopedia Anak	Various	Erlangga	Ensiklopedia	A2	Tersedia
13	Ensiklopedia Teknologi	Various	Erlangga	Ensiklopedia	A2	Tersedia
14	Ensiklopedia Hewan	Various	Gramedia	Ensiklopedia	A2	Tersedia
15	Ensiklopedia Flora	Various	Erlangga	Ensiklopedia	A2	Tersedia
16	Ensiklopedia Sejarah	Various	Gramedia	Ensiklopedia	A2	Tersedia
17	Ensiklopedia Sains	Various	Erlangga	Ensiklopedia	A2	Tersedia
18	Ensiklopedia Pendidikan	Various	Gramedia	Ensiklopedia	A2	Tersedia

Total: 70 buku
Tersedia: 65, Dipinjam: 3,
Rusak: 2, Hilang: 0

**Layanan Online Perpustakaan
Perpustakaan Mari Maca Kab. Karawang**

Tambah Buku **Edit Buku** **Hapus Buku** **Refresh**

Gambar 4 Tampilan daftar buku

Dashboard Admin - Sistem Perpustakaan Mari Maca

Admin

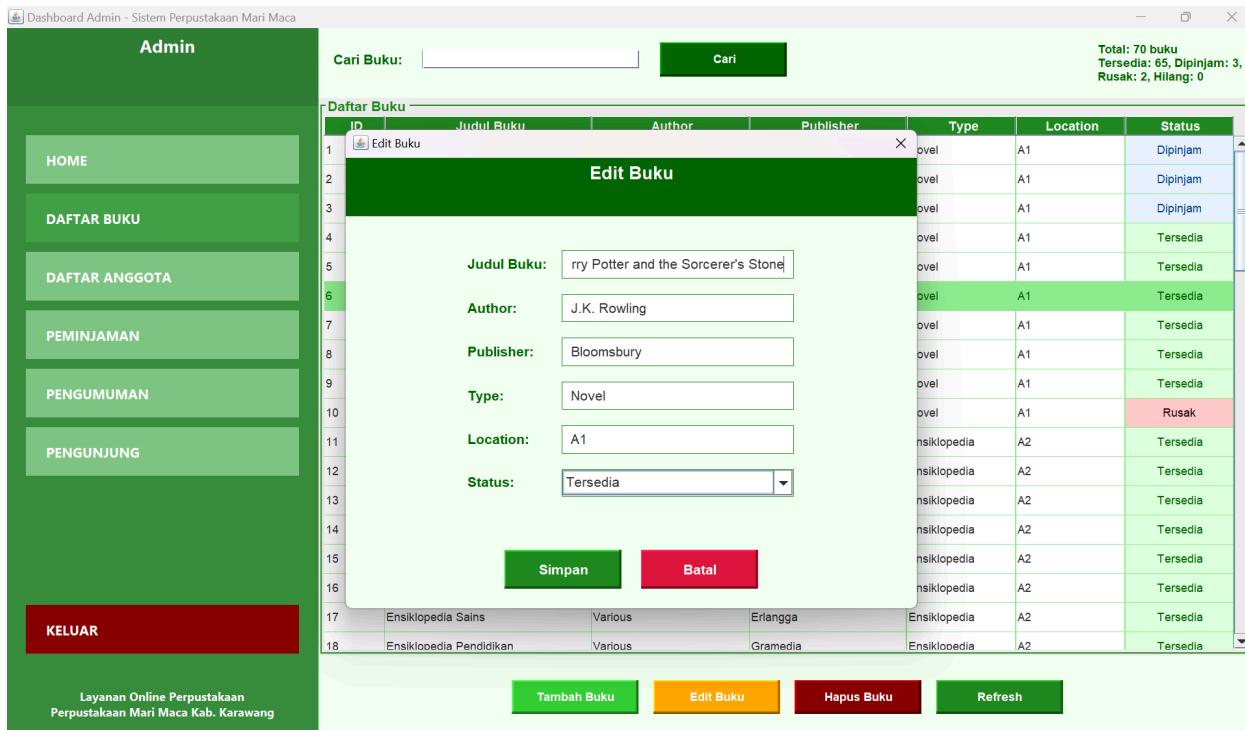
Daftar Buku						
ID	Judul Buku	Author	Publisher	Type	Location	Status
1	Tambah Buku Baru					
2						
3						
4						
5	Judul Buku:	<input type="text"/>				
6	Author:	<input type="text"/>				
7	Publisher:	<input type="text"/>				
8	Type:	<input type="text"/>				
9	Location:	<input type="text"/>				
10	Status:	<input type="text"/> Tersedia				
11	Simpan Batal					
12						
13						
14						
15						
16						
17	Ensiklopedia Sains	Various	Erlangga	Ensiklopedia	A2	Tersedia
18	Ensiklopedia Pendidikan	Various	Gramedia	Ensiklopedia	A2	Tersedia

Total: 70 buku
Tersedia: 65, Dipinjam: 3,
Rusak: 2, Hilang: 0

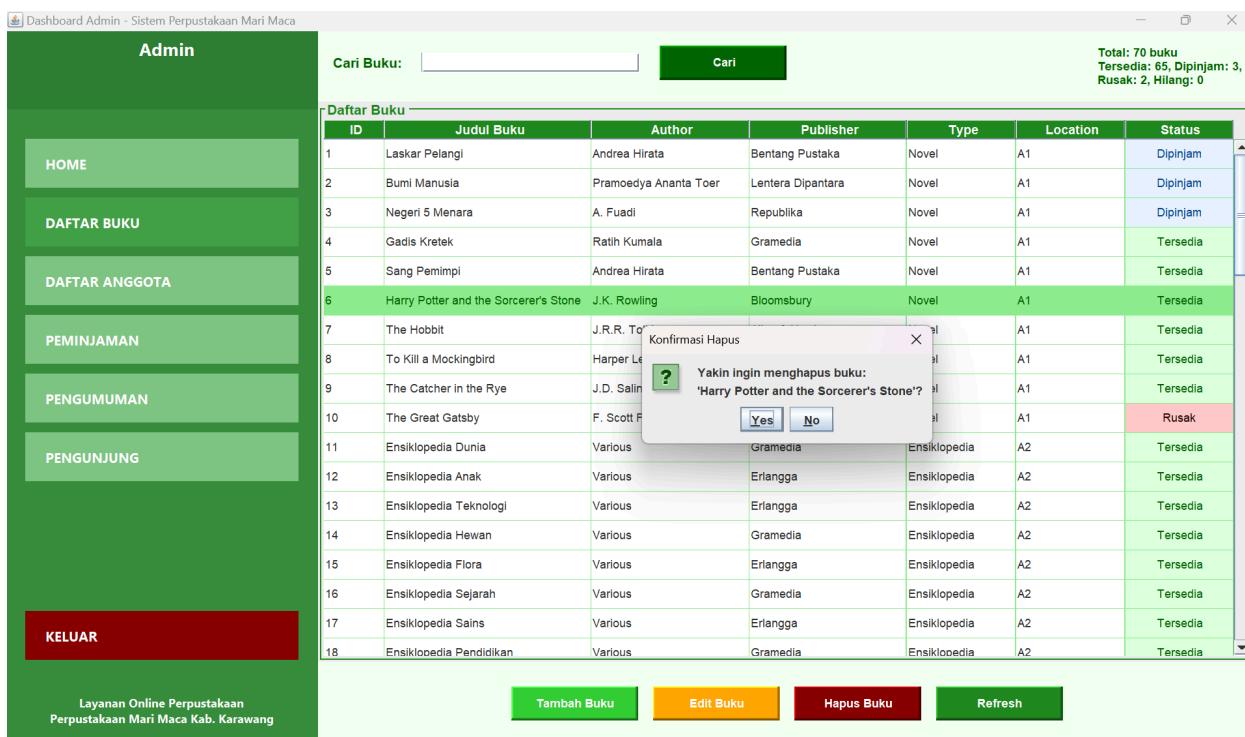
**Layanan Online Perpustakaan
Perpustakaan Mari Maca Kab. Karawang**

Tambah Buku **Edit Buku** **Hapus Buku** **Refresh**

Gambar 5 Tambah buku pada tampilan daftar buku



Gambar 6 Edit buku pada tampilan daftar buku



Gambar 7 Konfirmasi penghapusan daftar buku

Dashboard Admin - Sistem Perpustakaan Mari Maca

Admin

HOME	DAFTAR BUKU	DAFTAR ANGGOTA	PEMINJAMAN	PENGUMUMAN	PENGUNJUNG	KELUAR
Layanan Online Perpustakaan Perpustakaan Mari Maca Kab. Karawang						

Cari Anggota: Cari Total: 14 anggota | Aktif: 11 | Tidak Aktif: 3

Daftar Anggota

ID	Nama	Alamat	No_Telepon	Tanggal_Daftar	Email	Status_Aktif
1	Andi Wijaya	Jl. Melati No.10, Jakarta	081234567890	2023-01-05	andi.wijaya@example.com	Aktif
2	Siti Aminah	Jl. Mawar No.7, Depok	082345678901	2022-12-20	siti.aminah@example.com	Aktif
3	Budi Santoso	Jl. Kenanga No.20, Bogor	083456789012	2021-11-15	budi.santoso@example.com	Tidak Aktif
4	Dewi Lestari	Jl. Anggrek No.5, Tangerang	084567890123	2024-03-12	dewi.lestar@example.com	Aktif
5	Rian Pratama	Jl. Dahlia No.12, Bekasi	085678901234	2023-05-22	rian.pratama@example.com	Aktif
6	Nina Kartika	Jl. Melati No.3, Jakarta	086789012345	2020-09-09	nina.kartika@example.com	Tidak Aktif
7	Agus Salim	Jl. Flamboyan No.11, Depok	087890123456	2022-07-18	agus.salim@example.com	Aktif
8	Lina Marlina	Jl. Cempaka No.8, Bogor	088901234567	2023-04-30	lina.marlina@example.com	Aktif
9	Tono Susilo	Jl. Kamboja No.9, Tangerang	089012345678	2021-12-01	tono.susilo@example.com	Tidak Aktif
10	Fitri Handayani	Jl. Bougenvilla No.4, Bekasi	080123456789	2024-02-10	fitri.handayani@example.com	Aktif
11	asep imam wahyudi	JL. Candranala, Lebakwangi Kuningan	085889336609	2025-05-26	asep@gmail.com	Aktif
12	Muhammad Iqbal Pratama	Bekasi , Jawa Barat	0812345678	2025-05-27	iqbal@gmail.com	Aktif
13	Dery Husnairi	Cinagara, Kab Kuningan	0812345678	2025-06-07	dery@gmail.com	Aktif
14	Dila Septiola	Loji, Karawang	08654368886	2025-06-07	dila@gmail.com	Aktif

Tambah Anggota **Edit Anggota** **Hapus Anggota** **Cetak Kartu** **Refresh**

Gambar 8 Tampilan daftar anggota

Dashboard Admin - Sistem Perpustakaan Mari Maca

Admin

HOME	DAFTAR BUKU	DAFTAR ANGGOTA	PEMINJAMAN	PENGUMUMAN	PENGUNJUNG	KELUAR
Layanan Online Perpustakaan Perpustakaan Mari Maca Kab. Karawang						

Cari Anggota: Cari Total: 14 anggota | Aktif: 11 | Tidak Aktif: 3

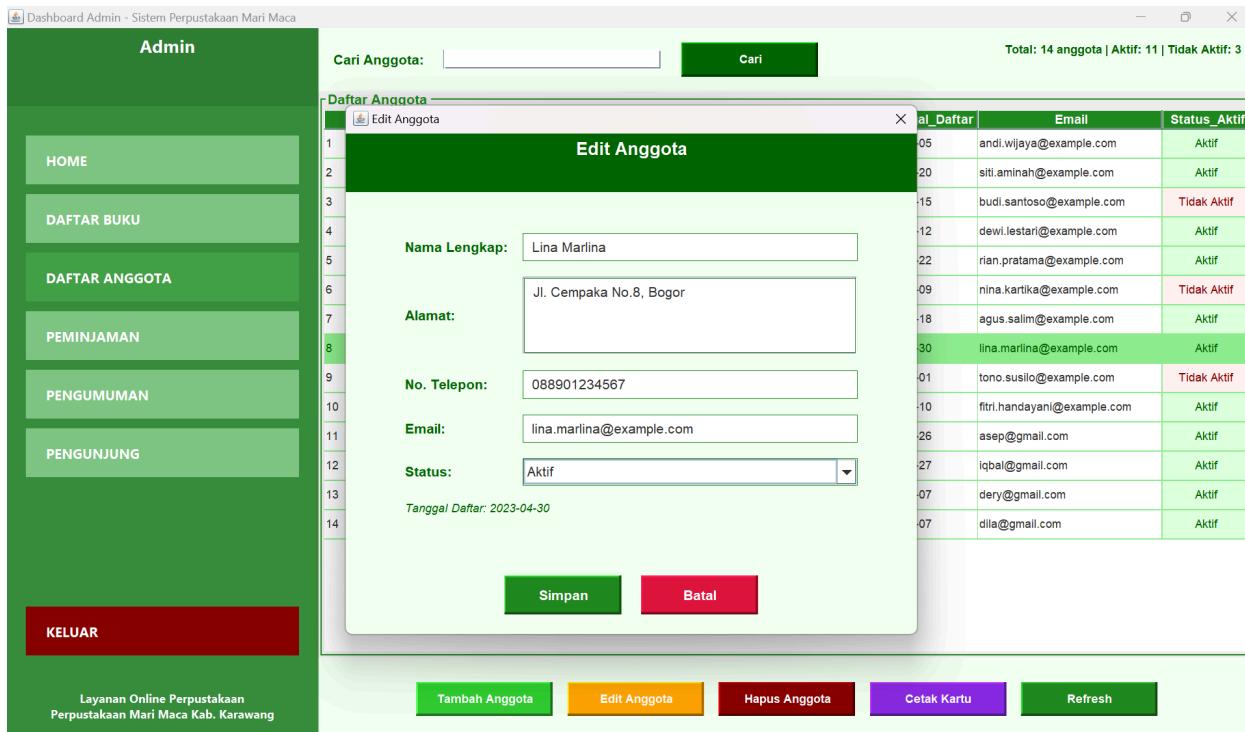
Daftar Anggota

Tambah Anggota Baru

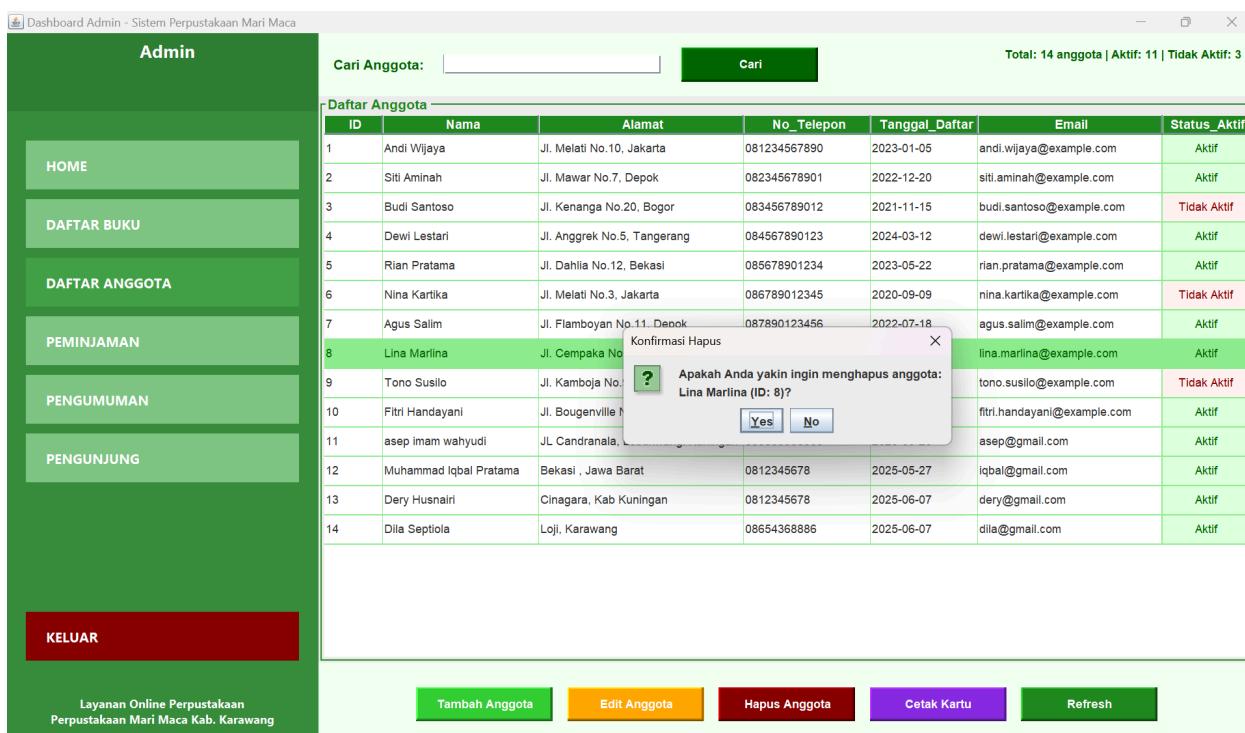
ID	al_Daftar	Email	Status_Aktif
1	05	andi.wijaya@example.com	Aktif
2	20	siti.aminah@example.com	Aktif
3	15	budi.santoso@example.com	Tidak Aktif
4	12	dewi.lestar@example.com	Aktif
5	22	rian.pratama@example.com	Aktif
6	09	nina.kartika@example.com	Tidak Aktif
7	18	agus.salim@example.com	Aktif
8	30	lina.marlina@example.com	Aktif
9	01	tono.susilo@example.com	Tidak Aktif
10	10	fitri.handayani@example.com	Aktif
11	26	asep@gmail.com	Aktif
12	27	iqbal@gmail.com	Aktif
13	07	dery@gmail.com	Aktif
14	07	dila@gmail.com	Aktif

Tambah Anggota **Edit Anggota** **Hapus Anggota** **Cetak Kartu** **Refresh**

Gambar 9 Tampilan tambah anggota baru



Gambar 10 Tampilan edit anggota



Gambar 11 Konfirmasi penghapusan anggota

The screenshot shows the 'Dashboard Admin - Sistem Perpustakaan Mari Maca' interface. On the left, a sidebar menu lists 'HOME', 'DAFTAR BUKU', 'DAFTAR ANGGOTA', 'PEMINJAMAN', 'PENGUMUMAN', 'PENGUNJUNG', and 'KELUAR'. Below the menu is a note: 'Layanan Online Perpustakaan Perpustakaan Mari Maca Kab. Karawang'. The main content area has a search bar with 'Cari Anggota:' and a 'Cari' button. A message at the top right says 'Total: 14 anggota | Aktif: 11 | Tidak Aktif: 3'. A table titled 'Daftar Anggota' lists 14 members with columns for ID, Name, Address, Phone Number, Registration Date, Email, and Status. Member 11, 'asep imam wahyudi', is highlighted in green. A modal window in the center displays a success message: 'Sukces Kartu anggota berhasil dicetak!' with an 'OK' button. At the bottom are buttons for 'Tambah Anggota', 'Edit Anggota', 'Hapus Anggota', 'Cetak Kartu' (highlighted in purple), and 'Refresh'.

Gambar 12 Pencetakan kartu anggota perpustakaan



Gambar 13 Kartu anggota perpustakaan

Dashboard Admin - Sistem Perpustakaan Mari Maca

Admin

- [HOME](#)
- [DAFTAR BUKU](#)
- [DAFTAR ANGGOTA](#)
- [PEMINJAMAN](#)
- [PENGUMUMAN](#)
- [PENGUNJUNG](#)
- [KELUAR](#)

Layanan Online Perpustakaan
Perpustakaan Mari Maca Kab. Karawang

Cari Anggota: **Cari** Total: 2 anggota | Aktif: 11 | Tidak Aktif: 3

Daftar Anggota

ID	Nama	Alamat	No_Telepon	Tanggal_Daftar	Email	Status_Aktif
1	Andi Wijaya	Jl. Melati No.10, Jakarta	081234567890	2023-01-05	andi.wijaya@example.com	Aktif
6	Nina Kartika	Jl. Melati No.3, Jakarta	086789012345	2020-09-09	nina.kartika@example.com	Tidak Aktif

Tambah Anggota **Edit Anggota** **Hapus Anggota** **Cetak Kartu** **Refresh**

Gambar 14 Pencarian Anggota

Dashboard Admin - Sistem Perpustakaan Mari Maca

Admin

- [HOME](#)
- [DAFTAR BUKU](#)
- [DAFTAR ANGGOTA](#)
- [PEMINJAMAN](#)
- [PENGUMUMAN](#)
- [PENGUNJUNG](#)
- [KELUAR](#)

Layanan Online Perpustakaan
Perpustakaan Mari Maca Kab. Karawang

MANAGEMENT PEMINJAMAN BUKU

Cari (Lilidul/Anggota/Status): **Cari** Total Peminjaman Aktif: 2

Tambah Peminjaman Baru

ID	Tgl Jatuh Tempo	Tgl Kembali	Status
10	27 May 2025	02 Jun 2025	Dikembalikan
9	-	-	Dipinjam
8	-	-	Dipinjam
7	-	-	Terlambat
5	27 May 2025	27 May 2025	Dikembalikan
4	27 May 2025	27 May 2025	Dikembalikan
2	27 May 2025	27 May 2025	Dikembalikan
1	27 May 2025	27 May 2025	Dikembalikan

Buku: **--- Pilih Buku ---**

Anggota: **--- Pilih Anggota ---**

Tgl Pinjam (YYYY-MM-DD):

Tgl Jatuh Tempo (YYYY-MM-DD):

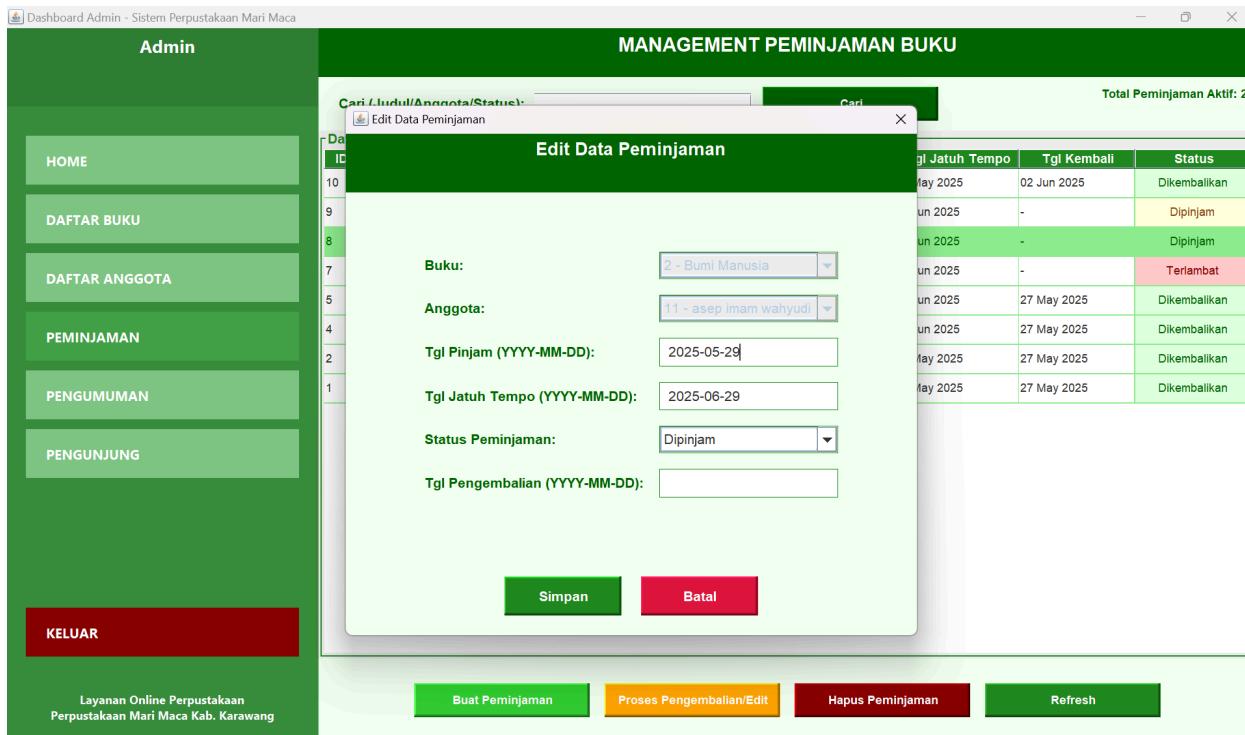
Status Peminjaman: **Dipinjam**

Tgl Pengembalian (YYYY-MM-DD):

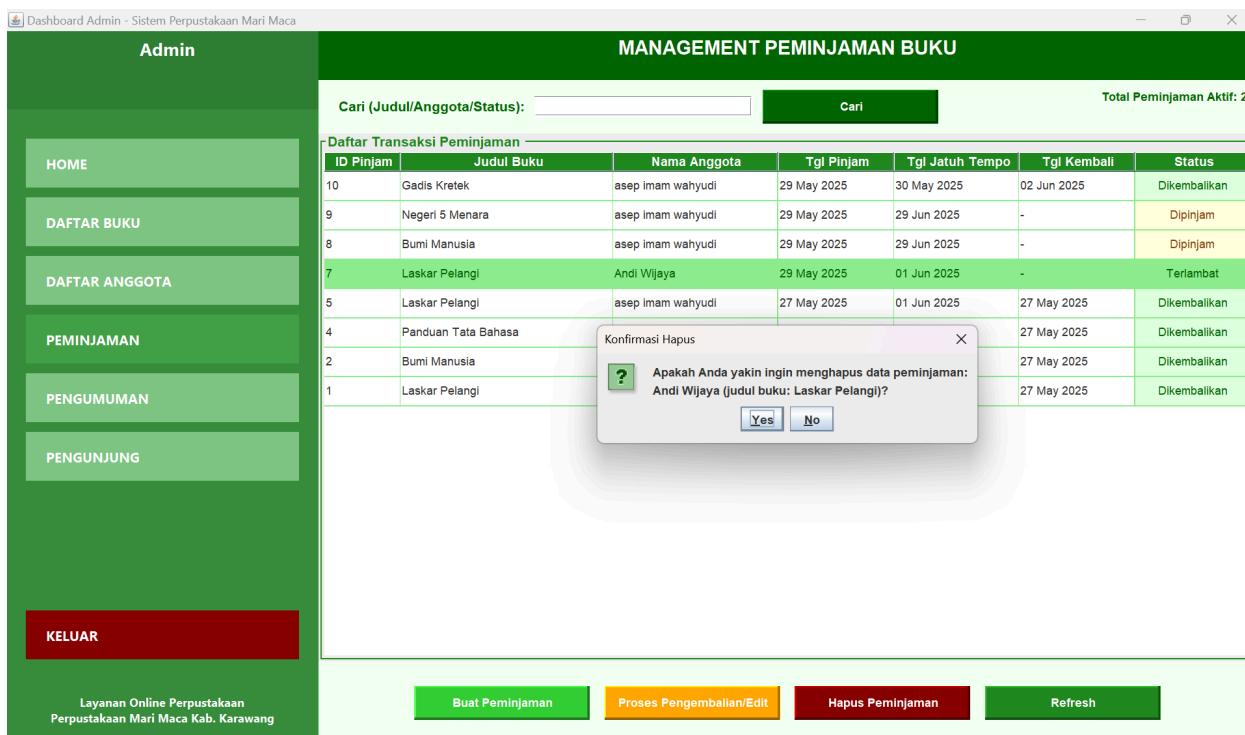
Simpan **Batal**

Buat Peminjaman **Proses Pengembalian/Edit** **Hapus Peminjaman** **Refresh**

Gambar 15 Tampilan tambah peminjaman buku



Gambar 16 Tampilan edit data peminjaman



Gambar 17 Konfirmasi penghapusan peminjaman

Dashboard Admin - Sistem Perpustakaan Mari Maca

Admin

- HOME**
- DAFTAR BUKU**
- DAFTAR ANGGOTA**
- PEMINJAMAN**
- PENGUMUMAN**
- PENGUNJUNG**

KELUAR

Layanan Online Perpustakaan
Perpustakaan Mari Maca Kab. Karawang

MANAGEMENT PEMINJAMAN BUKU

Cari (Judul/Anggota/Status): asep Hasil Pencarian: 5 transaksi ditemukan

ID Pinjam	Judul Buku	Nama Anggota	Tgl Pinjam	Tgl Jatuh Tempo	Tgl Kembali	Status
10	Gadis Kretek	asep imam wahyudi	29 May 2025	30 May 2025	02 Jun 2025	Dikembalikan
9	Negeri 5 Menara	asep imam wahyudi	29 May 2025	29 Jun 2025	-	Dipinjam
8	Bumi Manusia	asep imam wahyudi	29 May 2025	29 Jun 2025	-	Dipinjam
5	Laskar Pelangi	asep imam wahyudi	27 May 2025	01 Jun 2025	27 May 2025	Dikembalikan
2	Bumi Manusia	asep imam wahyudi	27 May 2025	29 May 2025	27 May 2025	Dikembalikan

Gambar 18 Tampilan pencarian peminjaman

Dashboard Admin - Sistem Perpustakaan Mari Maca

Admin

- HOME**
- DAFTAR BUKU**
- DAFTAR ANGGOTA**
- PEMINJAMAN**
- PENGUMUMAN**
- PENGUNJUNG**

KELUAR

Layanan Online Perpustakaan
Perpustakaan Mari Maca Kab. Karawang

MANAGEMENT PENGUMUMAN

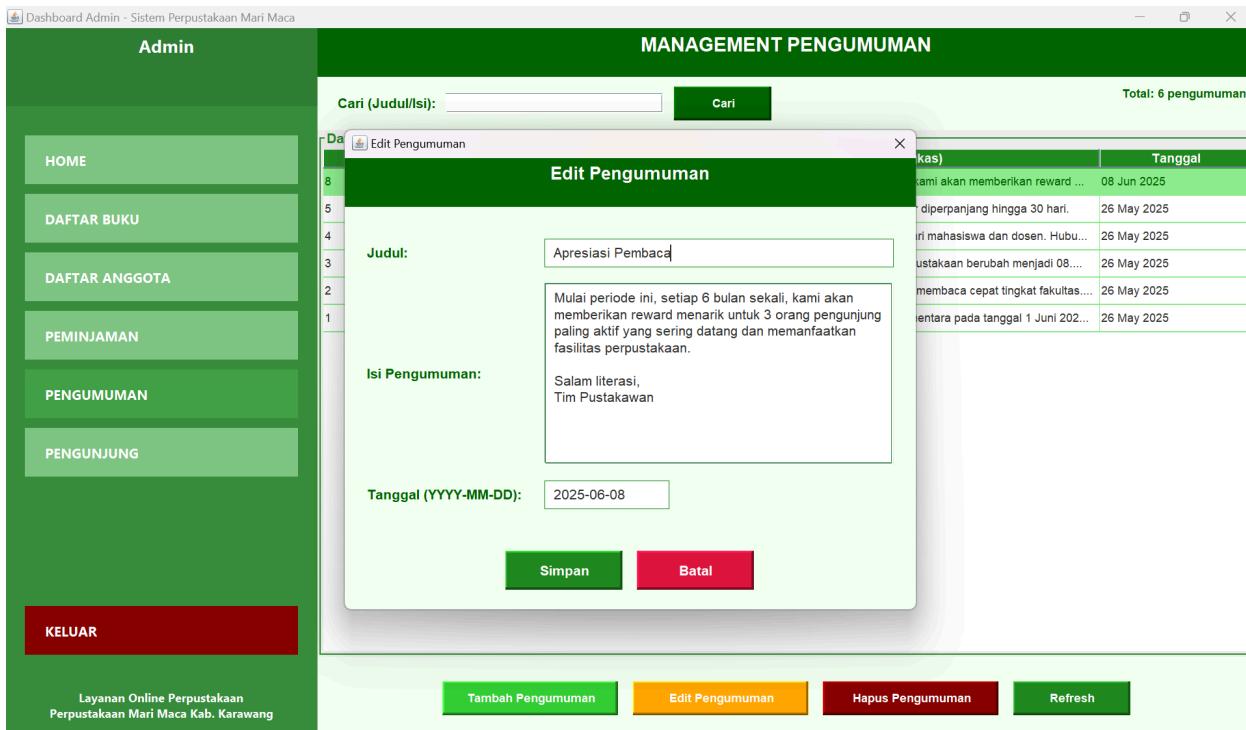
Cari (Judul/Isi): Total: 6 pengumuman

No	Judul	Isi Pengumuman	Tanggal
8	Tambah Pengumuman Baru
5
4
3
2
1

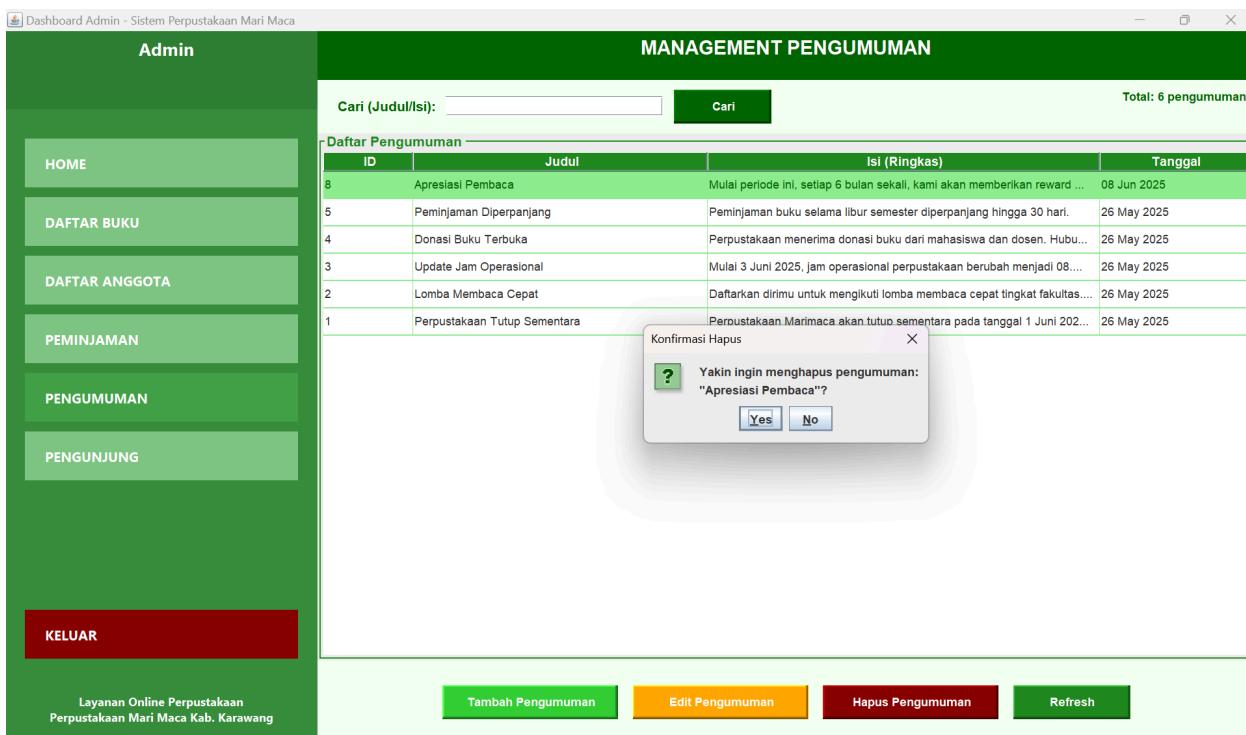
Tambah Pengumuman Baru

Judul:
Isi Pengumuman:
Tanggal (YYYY-MM-DD):

Gambar 19 Tampilan tambah pengumuman



Gambar 20 Tampilan edit pengumuman



Gambar 21 Konfirmasi pengumuman

Manajemen Data Pengunjung

Pencarian Berdasarkan Tanggal

ID Pengunjung	Nama	No Telepon	Tanggal Kunjungan
13	YUDI WAHYUDI	08977031709	2025-05-27
12	Nugrah kedua	081254475	2025-05-27
11	Nugrah Adinda	0812345678	2025-05-27
10	Fitri Handayani	080123456789	2025-05-10
9	Tono Susilo	089012345678	2025-05-09
8	Lina Marlina	088901234567	2025-05-08
7	Agus Salim	087890123456	2025-05-07
6	Nina Kartika	086789012345	2025-05-06
5	Rian Pratama	085678901234	2025-05-05
4	Dewi Lestari	084567890123	2025-05-04
3	Budi Santoso	083456789012	2025-05-03
2	Siti Aminah	082345678901	2025-05-02
1	Andi Wijaya	081234567890	2025-05-01

Total Pengunjung: 13

Gambar 22 Tampilan manajemen data pengunjung

Konfirmasi Keluar

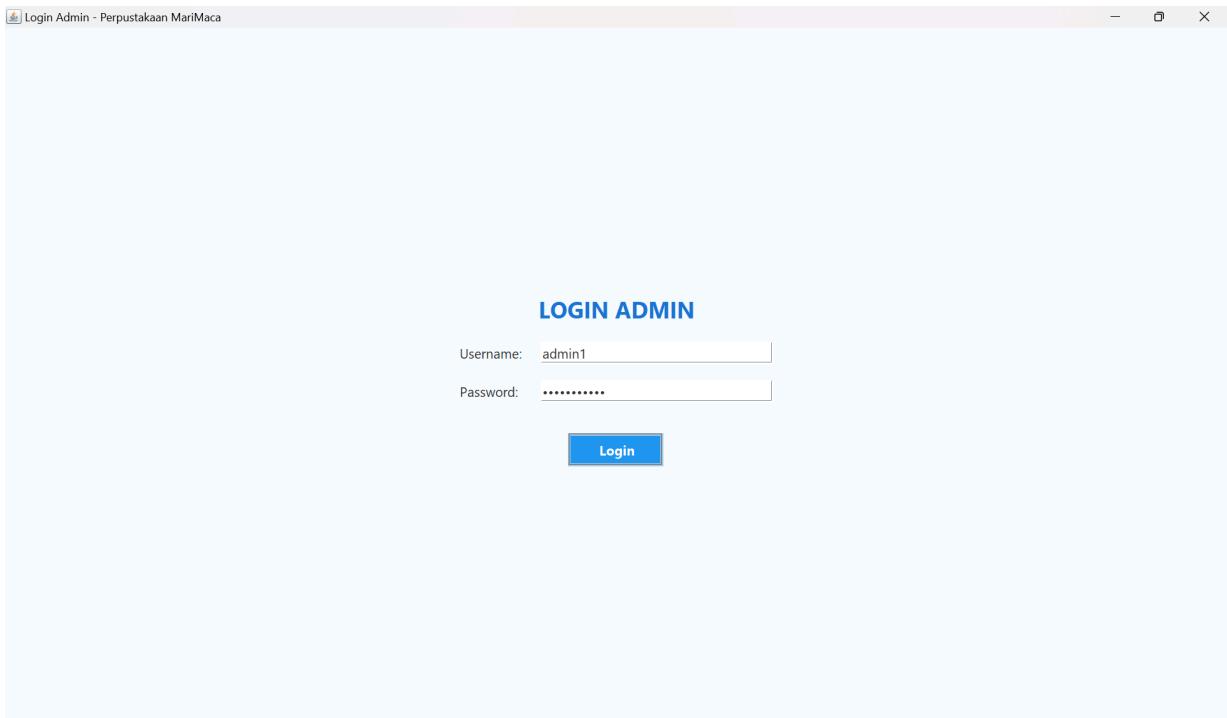
Apakah Anda yakin ingin keluar?

Yes No

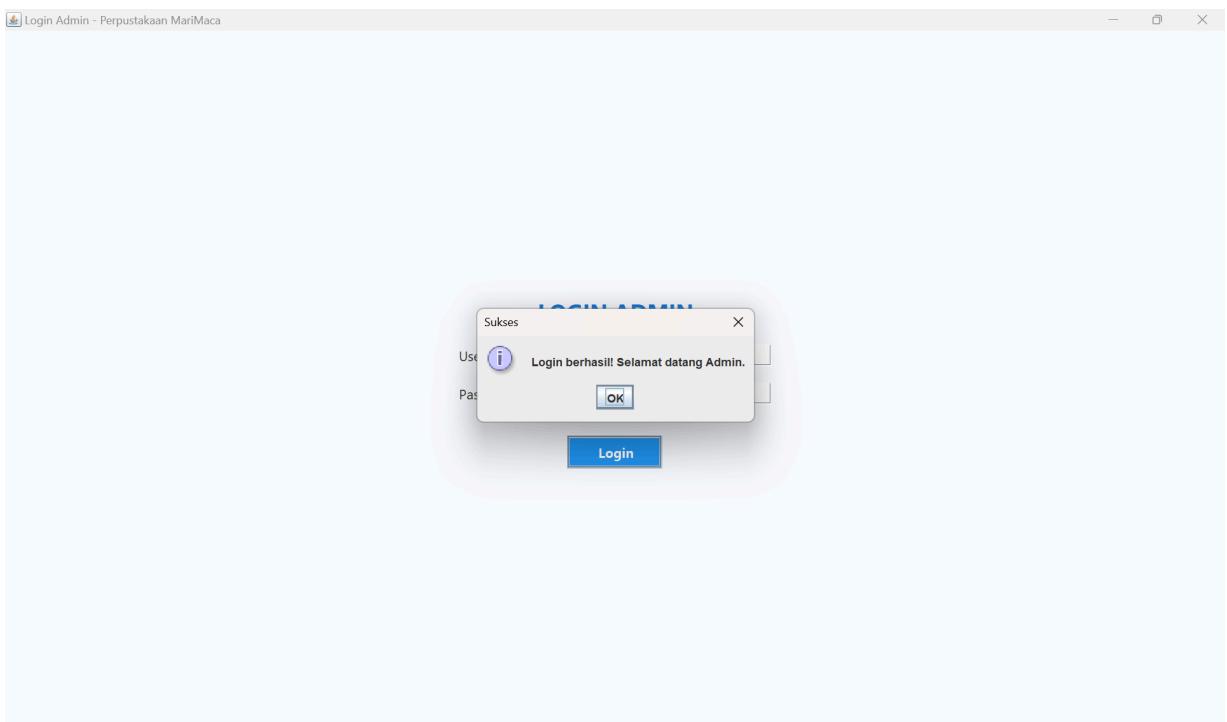
Selamat Menikmati Layanan Online

Perpustakaan Mari Maca

Gambar 23 Konfirmasi logout sistem admin

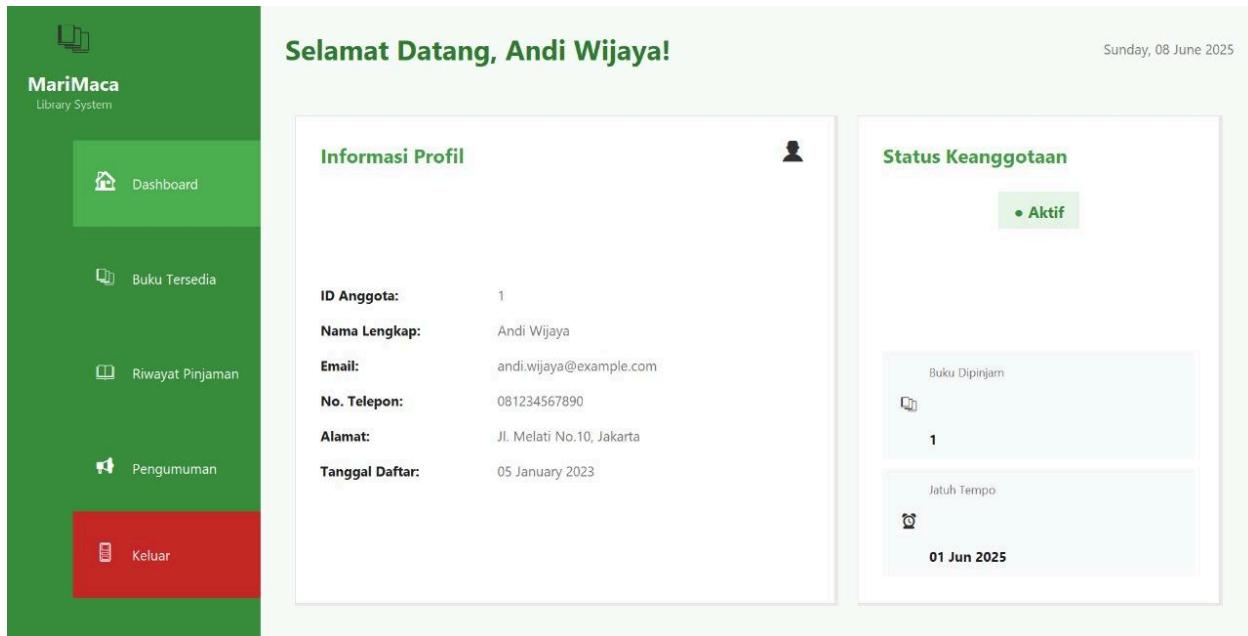


Gambar 24 Tampilan login by admin

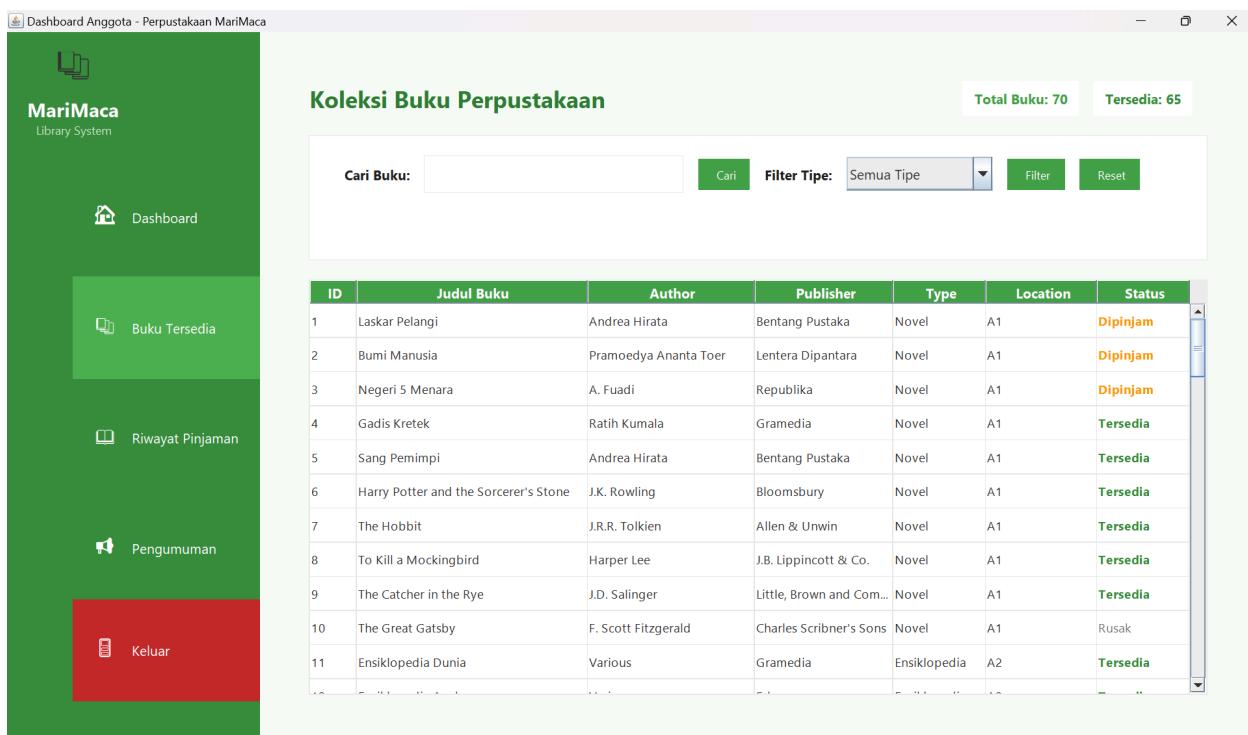


Gambar 25 Greeting login berhasil

2.6.2 Tampilan Anggota



Gambar 26 Dashboard Anggota Perpustakaan



Gambar 27 Panel Buku Perpustakaan

The screenshot shows the MariMac Library System dashboard. On the left sidebar, there are four main menu items: 'Dashboard' (green), 'Buku Tersedia' (green), 'Riwayat Pinjaman' (green), 'Pengumuman' (green), and 'Keluar' (red). The main content area is titled 'Koleksi Buku Perpustakaan'. It displays a search bar with 'Cari Buku:' and 'Filter Tipi: Sejarah', along with buttons for 'Cari', 'Filter', and 'Reset'. Above the search bar, it says 'Total Buku: 70' and 'Tersedia: 65'. A table lists 10 books, all marked as 'Tersedia' (Available). The columns are ID, Judul Buku, Author, Publisher, Type, Location, and Status.

ID	Judul Buku	Author	Publisher	Type	Location	Status
41	Sejarah Indonesia	Taufik Abdullah	Gramedia	Sejarah	A5	Tersedia
42	Sejarah Dunia Modern	Eric Hobsbawm	Penguin	Sejarah	A5	Tersedia
43	Sejarah Perang Dunia	Winston Churchill	Penguin	Sejarah	A5	Tersedia
44	Sejarah Asia	John Keay	HarperCollins	Sejarah	A5	Tersedia
45	Sejarah Amerika	Howard Zinn	Harper Perennial	Sejarah	A5	Tersedia
46	Sejarah Eropa	Norman Davies	Oxford University Press	Sejarah	A5	Tersedia
47	Sejarah Timur Tengah	Bernard Lewis	Oxford University Press	Sejarah	A5	Tersedia
48	Sejarah Afrika	John Iliffe	Cambridge University ...	Sejarah	A5	Tersedia
49	Sejarah Australia	Henry Reynolds	Allen & Unwin	Sejarah	A5	Tersedia
50	Sejarah Jepang	Marius B. Jansen	Harvard University Press	Sejarah	A5	Tersedia

Gambar 28 Tampilan Buku Tersedia

The screenshot shows the MariMac Library System dashboard. The left sidebar has the same four menu items: 'Dashboard', 'Buku Tersedia', 'Riwayat Pinjaman', 'Pengumuman', and 'Keluar'. The main content area is titled 'Koleksi Buku Perpustakaan'. It includes a search bar with 'Cari Buku: Indonesia' and 'Filter Tipi: Semua Tipi', with buttons for 'Cari', 'Filter', and 'Reset'. Above the search bar, it says 'Total Buku: 70' and 'Tersedia: 65'. A table lists 8 books found, all marked as 'Tersedia' (Available). The columns are ID, Judul Buku, Author, Publisher, Type, Location, and Status.

ID	Judul Buku	Author	Publisher	Type	Location	Status
41	Sejarah Indonesia	Taufik Abdullah	Gramedia	Sejarah	A5	Tersedia
61	Kamus Besar Bahasa Indonesia	Pusat Bahasa	Balai Pustaka	Referensi	A7	Tersedia
62	Kamus Inggris-Indonesia	John Smith	Oxford	Referensi	A7	Tersedia
66	Manual Bahasa Indonesia	Pusat Bahasa	Balai Pustaka	Referensi	A7	Tersedia
67	Tesaurus Bahasa Indonesia	Pusat Bahasa	Balai Pustaka	Referensi	A7	Tersedia
68	Ensiklopedia Indonesia	Various	Gramedia	Referensi	A7	Tersedia

Gambar 29 Tampilan Pencarian Ketersediaan Buku

Judul Buku	Tgl Pinjam	Jatuh Tempo	Tgl Kembali	Status
Laskar Pelangi	29 May 2025	01 Jun 2025	Belum Kembali	Dipinjam
Laskar Pelangi	27 May 2025	27 May 2025	27 May 2025	Dikembalikan

Gambar 30 Riwayat Peminjaman Anggota

Apresiasi Pembaca

Mulai periode ini, setiap 6 bulan sekali, kami akan memberikan reward menarik untuk 3 orang pengunjung paling aktif yang sering datang dan memanfaatkan fasilitas perpustakaan.

Salam literasi,
Tim Pustakawan

Dipublikasikan : 08 June 2025

Perpustakaan Tutup Sementara

Perpustakaan Marimaca akan tutup sementara pada tanggal 1 Juni 2025 karena maintenance sistem.

Dipublikasikan : 26 May 2025

Lomba Membaca Cepat

Daftarkan dirimu untuk mengikuti lomba membaca cepat tingkat fakultas. Pendaftaran dibuka hingga 10 Juni 2025.

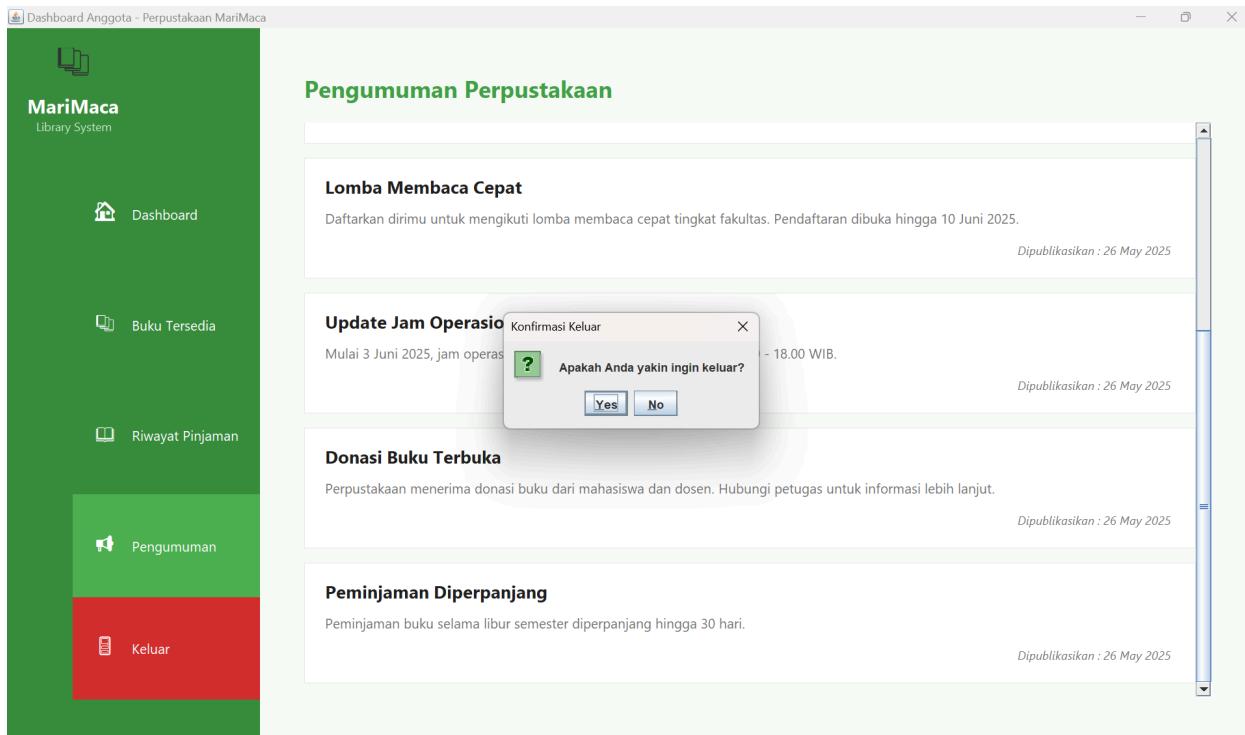
Dipublikasikan : 26 May 2025

Update Jam Operasional

Mulai 3 Juni 2025, jam operasional perpustakaan berubah menjadi 08.00 - 18.00 WIB.

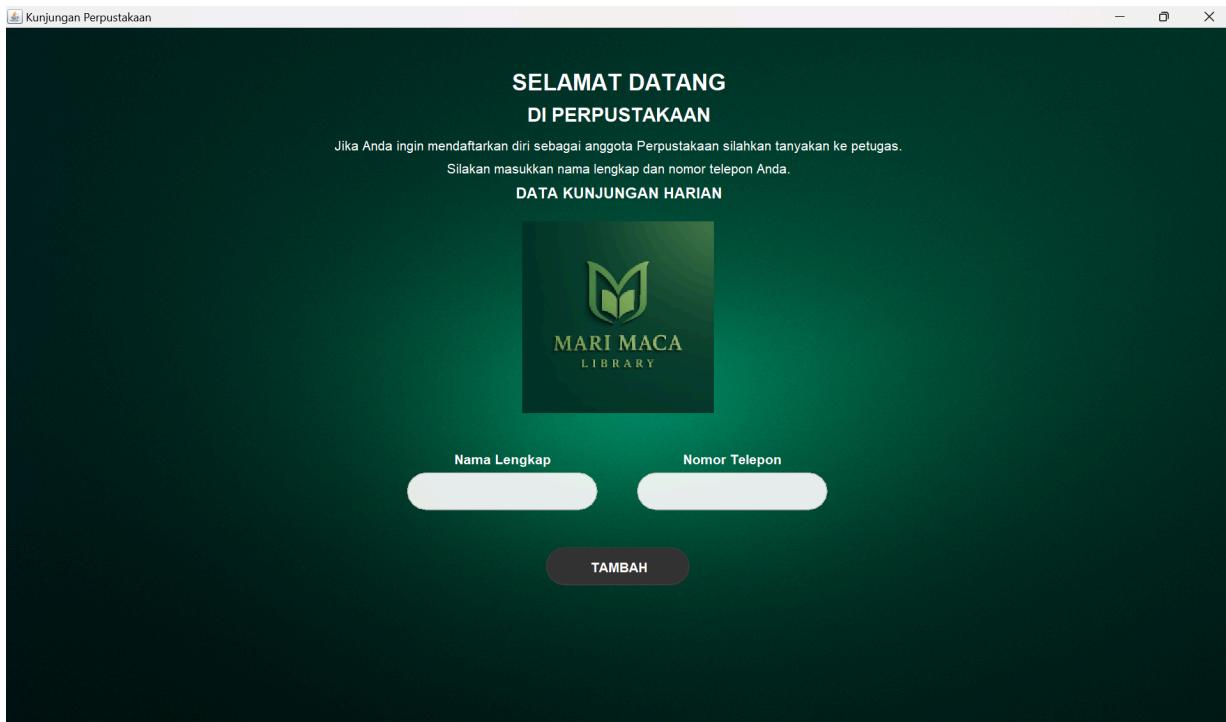
Dipublikasikan : 26 May 2025

Gambar 31 Pengumuman Informasi Perpustakaan



Gambar 32 Konfirmasi Logout Anggota

2.6.3 Tampilan Welcome



Gambar 33 Tampilan Pendataan Pengunjung

A screenshot of a Windows application window titled 'Dashboard Perpustakaan' under the 'MARIMACA' library. The dashboard features a green header bar with the library name and a search bar. Below the header, the title 'Dashboard Perpustakaan' is displayed. The main content area contains several cards with different types of information:

- Pengumuman**: Announcements.
 - Perpustakaan Marimaca akan tutup sementara pada tanggal 1 Juni 2025 karena maintenance sistem. (Date: 2025-05-26)
 - Daftarkan dirimu untuk mengikuti lomba membaca cepat tingkat fakultas. Pendaftaran dibuka hingga 10 Juni 2025. (Date: 2025-05-26)
- Update Jam Operasional**: Operational hours update.
 - Mulai 3 Juni 2025, jam operasional perpustakaan berubah menjadi 08.00 - 18.00 WIB. (Date: 2025-05-26)
- Donasi Buku Terbuka**: Book donation information.
 - Perpustakaan menerima donasi buku dari mahasiswa dan dosen. Hubungi petugas untuk informasi lebih lanjut. (Date: 2025-05-26)
- Peminjaman Diperpanjang**: Extended borrowing notice.
 - Peminjaman buku selama libur semester diperpanjang hingga 30 hari. (Date: 2025-05-26)
- Apresiasi Pembaca**: Reader appreciation information.
 - Mulai periode ini, setiap 6 bulan sekali, kami akan memberikan reward menarik untuk 3 orang pengunjung paling aktif yang sering datang dan memanfaatkan fasilitas perpustakaan.
Salam literasi,
Tim Pustakawan
2025-06-06

Gambar 34 Tampilan Pengumuman Perpustakaan

The screenshot shows the MARIMACA library management system interface. At the top, there is a green header bar with the title "MARIMACA". Below the header, the main title "Dashboard Perpustakaan" is displayed. A navigation bar at the top left includes tabs for "Pengumuman" and "Daftar Pustaka", with "Daftar Pustaka" being the active tab.

The main content area displays a grid of book entries, each in its own box:

- Laskar Pelangi**: Penulis: Andrea Hirata, Penerbit: Bentang Pustaka, Jenis: Novel, Lokasi: A1. Status: Dipinjam.
- Bumi Manusia**: Penulis: Pramoedya Ananta Toer, Penerbit: Lentera Dipantara, Jenis: Novel, Lokasi: A1. Status: Dipinjam.
- Negeri 5 Menara**: Penulis: A. Fuadi, Penerbit: Republika, Jenis: Novel, Lokasi: A1. Status: Dipinjam.
- Gadis Kretek**: Penulis: Ratih Kumala, Penerbit: Gramedia, Jenis: Novel, Lokasi: A1. Status: Tersedia.
- Sang Pemimpি**: Penulis: Andrea Hirata, Penerbit: Bentang Pustaka, Jenis: Novel, Lokasi: A1. Status: Tersedia.
- Harry Potter and the Sorcerer's Stone**: Penulis: J.K. Rowling, Penerbit: Bloomsbury, Jenis: Novel, Lokasi: A1. Status: Tersedia.
- The Hobbit**: Penulis: J.R.R. Tolkien, Penerbit: Allen & Unwin, Jenis: Novel, Lokasi: A1. Status: Tersedia.
- To Kill a Mockingbird**: Penulis: Harper Lee, Penerbit: J.B. Lippincott & Co., Jenis: Novel, Lokasi: A1. Status: Tersedia.
- The Catcher in the Rye**: Penulis: J.D. Salinger, Penerbit: Little, Brown and Company, Jenis: Novel, Lokasi: A1. Status: Tersedia.
- The Great Gatsby**: Penulis: F. Scott Fitzgerald, Penerbit: Charles Scribner's Sons, Jenis: Novel, Lokasi: A1. Status: Tersedia.
- Ensiklopedia Dunia**: Penulis: Various, Penerbit: Gramedia, Jenis: Ensiklopedia, Lokasi: A2. Status: Tersedia.
- Ensiklopedia Anak**: Penulis: Various, Penerbit: Erlangga, Jenis: Ensiklopedia, Lokasi: A2. Status: Tersedia.

Gambar 35 Tampilan Daftar Pustaka

The screenshot shows the MARIMACA library management system interface. At the top, there is a green header bar with the title "MARIMACA". Below the header, the main title "Dashboard Perpustakaan" is displayed. A search bar at the top right contains the text "A7". A navigation bar at the top left includes tabs for "Pengumuman" and "Daftar Pustaka", with "Daftar Pustaka" being the active tab.

The main content area displays a grid of book entries, each in its own box:

- Kamus Besar Bahasa Indonesia**: Penulis: Pusat Bahasa, Penerbit: Balai Pustaka, Jenis: Referensi, Lokasi: A7. Status: Tersedia.
- Kamus Inggris-Indonesia**: Penulis: John Smith, Penerbit: Oxford, Jenis: Referensi, Lokasi: A7. Status: Tersedia.
- Glosarium Istilah IT**: Penulis: Tech Writer, Penerbit: Informatika, Jenis: Referensi, Lokasi: A7. Status: Tersedia.
- Daftar Singkatan**: Penulis: Various, Penerbit: Gramedia, Jenis: Referensi, Lokasi: A7. Status: Tersedia.
- Panduan Penulisan**: Penulis: Stephen King, Penerbit: Scribner, Jenis: Referensi, Lokasi: A7. Status: Tersedia.
- Manual Bahasa Indonesia**: Penulis: Pusat Bahasa, Penerbit: Balai Pustaka, Jenis: Referensi, Lokasi: A7. Status: Tersedia.
- Tesaurus Bahasa Indonesia**: Penulis: Pusat Bahasa, Penerbit: Balai Pustaka, Jenis: Referensi, Lokasi: A7. Status: Tersedia.
- Ensiklopedia Indonesia**: Penulis: Various, Penerbit: Gramedia, Jenis: Referensi, Lokasi: A7. Status: Tersedia.
- Atlas Dunia**: Penulis: National Geographic, Penerbit: National Geographic Society, Jenis: Referensi, Lokasi: A7.
- Panduan Tata Bahasa**: Penulis: Linguist A, Penerbit: Universitas Indonesia, Jenis: Referensi, Lokasi: A7.

Gambar 36 Pencarian Pustaka

2.7 Pengujian

Pengujian dilakukan untuk memastikan bahwa setiap fungsionalitas sistem berjalan sesuai dengan perancangan. Metode pengujian yang digunakan adalah Black Box Testing, dimana pengujian difokuskan pada verifikasi *input* dan *output* dari setiap fitur tanpa melihat struktur kode internalnya.

Berikut adalah skenario pengujian yang dilakukan pada setiap modul:

2.7.1 Pengujian Modul Admin

Kasus Uji	Skenario Pengujian	Langkah-Langkah	Hasil yang Diharapkan
TC-01	Manajemen Data Buku (CRUD)	Admin pertama-tama menekan tombol "Tambah Buku" untuk memunculkan BookDialog, kemudian mengisi semua field dan menyimpannya. Selanjutnya, admin memilih sebuah data buku dari tabel untuk diuji, menekan tombol "Edit Buku" untuk mengubah informasinya, dan terakhir memilih data buku lainnya untuk dihapus melalui tombol "Hapus Buku".	Data buku berhasil ditambahkan, diperbarui, dan dihapus. Tabel diperbarui secara otomatis setelah setiap operasi.
TC-02	Manajemen Data Anggota dan Cetak Kartu	Admin melakukan serangkaian operasi CRUD pada data anggota. Setelah itu, admin memilih salah satu data anggota dari tabel dan melanjutkan dengan menekan tombol "Cetak Kartu" untuk menguji fungsionalitas pencetakan.	Data anggota berhasil dikelola. Jendela dialog pencetakan muncul dan kartu anggota berhasil dicetak sesuai format.

TC-03	Manajemen Peminjaman	<p>Admin menekan tombol "Buat Peminjaman" untuk membuka PeminjamanDialog. Di dalam dialog tersebut, admin memverifikasi bahwa daftar pilihan hanya menampilkan buku yang tersedia dan anggota yang aktif. Setelah itu, admin menyimpan transaksi baru dan kemudian memproses pengembalian pada transaksi yang sudah ada untuk menguji alur lengkapnya.</p>	Transaksi berhasil dibuat dan status buku berubah menjadi "Dipinjam". Setelah pengembalian, status buku kembali "Tersedia".
TC-04	Manajemen Pengumuman (CRUD)	<p>Admin menguji seluruh fungsionalitas CRUD dengan melakukan operasi tambah, ubah, dan hapus pada data pengumuman melalui panel yang disediakan.</p>	Data pengumuman berhasil dikelola. Pengumuman yang dipublikasikan muncul di dasbor publik.

2.7.2 Pengujian Modul Anggota

Kasus Uji	Skenario Pengujian	Langkah-Langkah	Hasil yang Diharapkan
TC-05	Registrasi dan Login	<p>Skenario pengujian dimulai dengan mencoba mendaftar menggunakan ID Anggota yang tidak valid untuk memastikan sistem menolak. Kemudian, pendaftaran dilanjutkan dengan ID yang valid. Terakhir, pengguna melakukan login menggunakan kredensial dari akun yang baru saja berhasil dibuat.</p>	Pendaftaran gagal saat ID tidak valid. Pendaftaran berhasil saat ID valid. Login berhasil dan mengarahkan ke dasbor anggota.
TC-06	Dasbor dan Katalog	<p>Setelah login, anggota memverifikasi kebenaran data profil yang ditampilkan pada dasbor utama. Selanjutnya, anggota berpindah ke panel</p>	Data profil sesuai dengan database. Fitur pencarian dan filter pada katalog buku berfungsi dengan benar.

		katalog buku untuk menguji fungsi pencarian dan filter yang tersedia.	
--	--	---	--

2.7.3 Pengujian Modul Pengunjung

Kasus Uji	Skenario Pengujian	Langkah-Langkah	Hasil yang Diharapkan
TC-07	Pencatatan Kunjungan	Saat pertama kali membuka aplikasi, pengunjung langsung diarahkan untuk mengisi nama dan nomor telepon pada formulir yang tersedia, kemudian menekan tombol "TAMBAH" untuk menyimpan data kunjungannya.	Data kunjungan berhasil tersimpan di basis data. Halaman berpindah ke dasbor informasi publik.
TC-08	Akses Informasi Publik	Setelah masuk ke dasbor publik, pengguna melihat informasi yang ditampilkan seperti daftar buku dan pengumuman, kemudian melanjutkan dengan menggunakan fitur pencarian untuk menemukan data spesifik.	Daftar buku dan pengumuman berhasil ditampilkan. Fungsi pencarian memberikan hasil yang relevan.

2.8 Hasil Pengujian

Berdasarkan serangkaian skenario pengujian yang telah dilaksanakan, dapat disimpulkan bahwa seluruh fungsionalitas utama pada Sistem Informasi Perpustakaan "Mari Maca" telah berjalan sesuai dengan perancangan dan kebutuhan fungsional. Sistem mampu menangani operasi data secara akurat, mengelola alur kerja antar modul, dan menyajikan informasi dengan benar kepada setiap jenis pengguna.

BAB III

PENUTUP

3.1 Kesimpulan

Sistem perpustakaan berbasis Graphical User Interface (GUI) yang telah dikembangkan menggunakan Java Swing ini mampu memberikan kemudahan dan efisiensi dalam pengelolaan data perpustakaan secara digital. Sistem ini dirancang untuk mendukung aktivitas utama perpustakaan seperti pencatatan daftar buku, proses peminjaman, pengembalian, serta pengumuman informasi perpustakaan yang bisa dijangkau oleh para anggota perpustakaan.

Penggunaan konsep peran pengguna, yaitu admin, anggota, dan pengunjung, memberikan kejelasan dalam pembagian hak akses dan tanggung jawab. Hal ini tidak hanya meningkatkan keamanan sistem, tetapi juga mendukung pengelolaan yang lebih tertib dan terorganisir.

UI pengguna yang sederhana dan intuitif memungkinkan sistem ini dioperasikan dengan mudah oleh pengguna dari berbagai kalangan, tanpa memerlukan latar belakang teknis yang mendalam.

Secara keseluruhan, pengembangan sistem ini menunjukkan bahwa teknologi GUI dapat dimanfaatkan secara optimal untuk menciptakan solusi digital yang efisien, efektif, dan mudah digunakan dalam bidang pengelolaan perpustakaan.