LAPORAN PROYEK APLIKASI MAHASISWA

"*StatCredit*"

Dosen Pengampu:
Ibnu Santoso, S.S.T., M.T.

Disusun Oleh:
Kelompok 2
2KS2

Ahmad Husein Nasution	(222312952)
Muhammad Muhlis Aditya Nur Wahid	(222313249)
M. Rezky Raya Kilwouw	(222313190)
Zakia Faza Adila	(222313443)

PROGRAM STUDI D-IV KOMPUTASI STATISTIK
POLITEKNIK STATISTIKA STIS
2024/2025

DAFTAR ISI

KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Allah Subhanahu Wata'ala, karena atas rahmat dan karuniaNya, kami dapat menyelesaikan Laporan Kemajuan Kedua untuk proyek pengembangan aplikasi desktop "StatCredit". Laporan ini disusun untuk memenuhi salah satu tugas mata kuliah Proyek Aplikasi Terapan.

StatCredit merupakan aplikasi yang dirancang untuk mengelola Credit mahasiswa Politeknik Statistika STIS, yang diperoleh dari berbagai kegiatan positif. Aplikasi ini diharapkan dapat menjadi sistem yang terpusat dan transparan untuk mencatat, memvalidasi, dan memanfaatkan Credit tersebut.

Dalam laporan kemajuan kedua ini, kami memfokuskan pembahasan pada aspek perancangan dan pengembangan antarmuka pengguna (GUI) menggunakan JavaFX serta implementasi mekanisme *event-handling* yang menjadi dasar interaksi pengguna dengan sistem.

Kami mengucapkan terima kasih kepada Bapak Ibnu Santoso, S.S.T., M.T., selaku dosen pengampu yang telah memberikan bimbingan. Kami menyadari laporan ini masih memiliki kekurangan, oleh karena itu kritik dan saran yang membangun sangat kami harapkan. Semoga laporan ini dapat memberikan gambaran yang jelas mengenai kemajuan proyek yang telah kami kerjakan.

Jakarta Timur, 20 Juli 2025

Tim Penyusun

BAB I
PENDAHULUAN

1.1. Latar Belakang

Kehidupan mahasiswa di Politeknik Statistika STIS tidak hanya terbatas pada kegiatan akademik di dalam kelas. Banyak mahasiswa yang aktif berpartisipasi dalam berbagai kegiatan non-akademik seperti kejuaraan, kepanitiaan dalam organisasi, serta mengikuti workshop untuk pengembangan diri. Namun, seringkali pencatatan prestasi dan keaktifan ini tidak terpusat, sehingga sulit untuk diukur dan diapresiasi secara formal oleh institusi.

Untuk menjawab tantangan tersebut, diperlukan sebuah sistem yang dapat menguantifikasi dan mengelola kontribusi mahasiswa secara komprehensif. StatCredit dikembangkan sebagai solusi berupa aplikasi desktop yang berfungsi untuk mencatat Credit yang diperoleh mahasiswa dari empat kategori utama: akademik, kejuaraan, organisasi, dan workshop. Credit ini tidak hanya menjadi bukti keaktifan, tetapi juga memiliki nilai praktis. Mahasiswa dapat menggunakannya sebagai modal dalam lelang penempatan kerja atau menukarkannya dengan berbagai barang yang tersedia di koperasi mahasiswa.

Aplikasi ini dirancang dengan dua hak akses utama: Mahasiswa sebagai pengguna dan Admin sebagai pengelola sistem, untuk memastikan data yang valid dan terkelola dengan baik.

1.2. Tujuan

Tujuan dari pengembangan proyek StatCredit adalah:

1. Merancang dan membangun aplikasi desktop fungsional menggunakan teknologi JavaFX untuk manajemen Credit mahasiswa.
2. Menyediakan platform bagi mahasiswa untuk memantau perolehan dan penggunaan Credit mereka secara transparan.
3. Mengembangkan fitur bagi mahasiswa untuk menukarkan (redeem) Credit dengan barang di koperasi mahasiswa.
4. Membangun antarmuka khusus untuk admin guna melakukan validasi, input data Credit, dan pengelolaan sistem secara efisien.
5. Secara spesifik pada laporan ini, mendokumentasikan progres pengembangan antarmuka grafis (GUI) dan implementasi *event-handling* sebagai fondasi interaksi aplikasi.

1.3. Manfaat

Manfaat dari pengembangan proyek StatCredit adalah:

1. Bagi Mahasiswa: Menyediakan cara yang jelas dan terpusat untuk melacak pencapaian non-akademik, serta memberikan insentif nyata melalui penukaran Credit dan keuntungan saat lelang penempatan.
2. Bagi Admin/Institusi: Memudahkan proses administrasi dan validasi keaktifan mahasiswa, menciptakan sistem penghargaan yang transparan, dan menyediakan data agregat mengenai keterlibatan mahasiswa.
3. Bagi Koperasi Mahasiswa: Terintegrasi dengan sistem Credit, berpotensi meningkatkan partisipasi dan transaksi di koperasi.

1.4. Ruang Lingkup Proyek

Ruang lingkup proyek ini mencakup:

1. Pengembangan aplikasi adalah untuk platform Desktop menggunakan JavaFX.
2. Fitur untuk Mahasiswa: Login, melihat total Credit, melihat riwayat transaksi Credit (pendapatan dan pengeluaran), melihat katalog barang, dan melakukan penukaran barang.
3. Fitur untuk Admin: Login, mengelola data mahasiswa, melakukan validasi dan input Credit untuk mahasiswa, serta mengelola data barang di katalog koperasi.
4. Sistem hanya mencakup proses pengajuan dan pengelolaan pinjaman. Aspek pendanaan (sumber dana) tidak termasuk dalam lingkup teknis proyek ini.
5. Pengguna aplikasi adalah mahasiswa aktif Politeknik Statistika STIS.

BAB II
STUDI KELAYAKAN

2.1. Kebutuhan Bisnis

Kebutuhan utama yang ingin dijawab adalah inefisiensi dalam sistem pencatatan dan apresiasi kegiatan mahasiswa yang masih manual dan tersebar. Institusi memerlukan alat yang efektif untuk memotivasi mahasiswa agar aktif di berbagai bidang dan secara bersamaan memiliki data yang terstruktur mengenai hal tersebut. Sistem Credit ini menjadi "gamifikasi" dari produktivitas mahasiswa, yang membutuhkan sebuah aplikasi khusus agar dapat berjalan secara efisien, transparan, dan bebas dari kesalahan manusia.

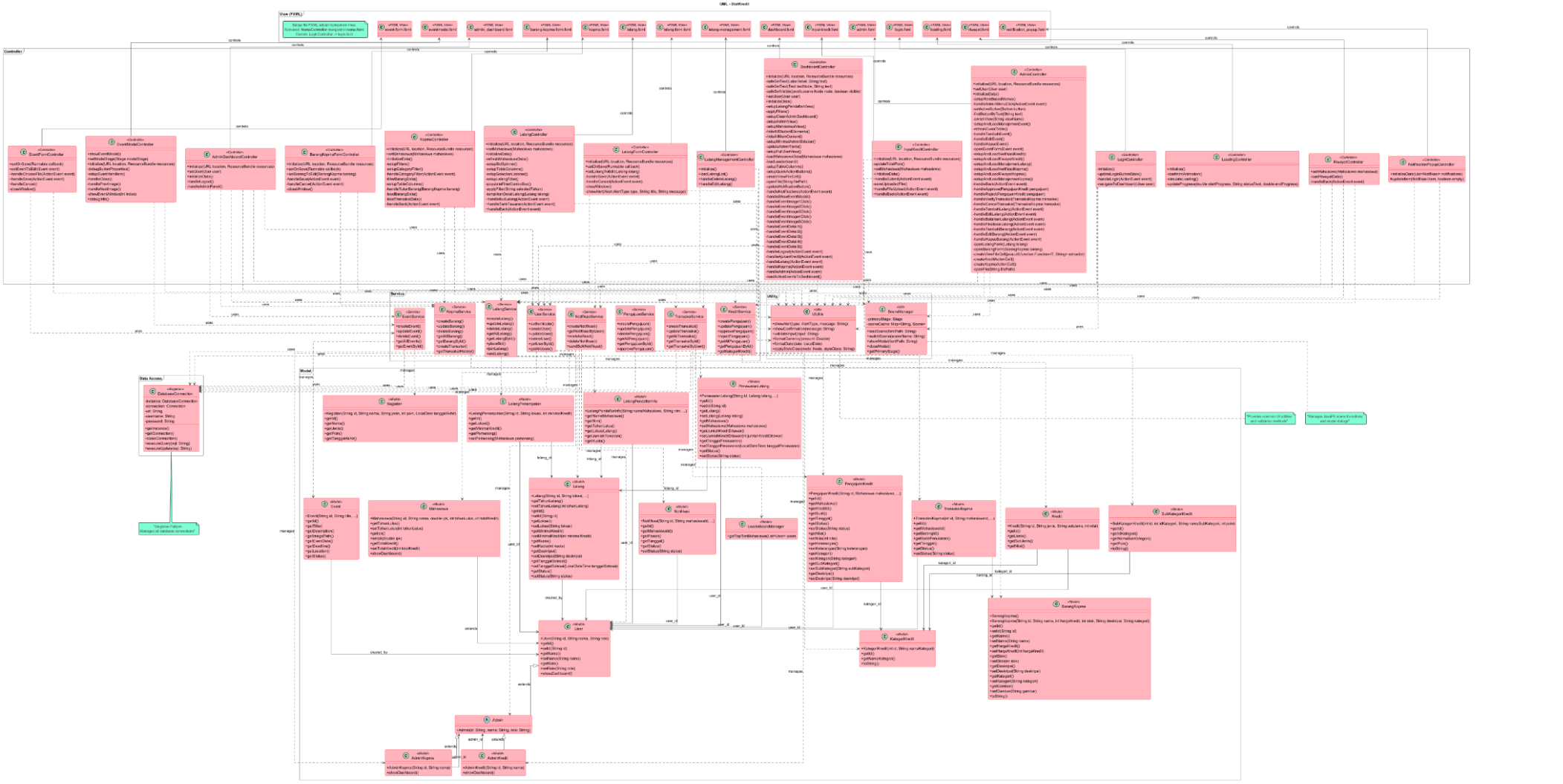
2.2. Hasil Akhir Bisnis

Hasil akhir yang dituju adalah sebuah produk perangkat lunak (aplikasi desktop) bernama StatCredit yang siap diimplementasikan di lingkungan Politeknik Statistika STIS. Aplikasi ini akan menjadi alat bantu utama bagi admin dalam mengelola Credit dan menjadi platform interaktif bagi mahasiswa untuk melihat serta memanfaatkan hasil kerja keras mereka.

BAB III
PERANCANGAN SISTEM

3.1. Penjelasan Program

StatCredit adalah aplikasi desktop yang dibangun dengan Java dan platform JavaFX. Arsitektur program dirancang menggunakan pola desain *Model-View-Controller* (MVC) untuk arsitektur utama yang memisahkan data, tampilan, dan logika kontrol; *Layered Architecture* yang membagi sistem menjadi lapisan-lapisan independen (*View, Controller, Service, Data Access*) untuk meningkatkan modularitas, *Service Layer* supaya dapat mengisolasi logika bisnis dari *Controller*, membuatnya lebih mudah diuji dan dikelola; *Singleton Pattern* yang digunakan pada DatabaseConnection untuk memastikan hanya ada satu titik akses ke database; dan *Data Access Object* (DAO): Lapisan Service dan Data Access secara bersama-sama membentuk pola DAO, di mana detail persistensi data disembunyikan dari seluruh aplikasi.


UML - StaffKredit
View (FXML)
Controller
Service
Model
Data Access
Utility

3.2. Penjelasan Komponen Utama

3.2.1. Layer Model

Lapisan ini adalah jantung dari aplikasi yang merepresentasikan data dan logika bisnis inti.

- **Tujuan**: Berisi kelas-kelas yang mendefinisikan objek-objek data (entitas) dalam sistem, seperti Mahasiswa, Lelang, PengajuanKredit, dll. Kelas-kelas ini sering disebut sebagai POJO (Plain Old Java Object).
- **Kelas Utama**:
    - **User**: Kelas *abstract* yang menjadi dasar bagi semua jenis pengguna.
    - **Mahasiswa**, **AdminKopma**, **AdminKredit**: Kelas turunan dari User yang merepresentasikan peran spesifik.
    - **Lelang**, **Event**, **BarangKopma**: Entitas utama yang dikelola dalam sistem.
    - **PengajuanKredit**, **TransaksiKopma**, **PenawaranLelang**: Entitas yang merepresentasikan transaksi atau interaksi dalam sistem.
- **Relasi**: Terdapat relasi warisan (inheritance) di mana Mahasiswa dan Admin adalah turunan dari User. Selain itu, terdapat banyak relasi asosiasi, misalnya sebuah PengajuanKredit dimiliki oleh satu Mahasiswa.

3.2.2 Layer View (FXML)

Lapisan ini bertanggung jawab penuh atas presentasi dan antarmuka pengguna (UI).

- **Tujuan**: Menampilkan data kepada pengguna dan menangkap input dari pengguna.
- **Teknologi**: Dibangun menggunakan file **FXML**, yang merupakan format berbasis XML untuk mendefinisikan struktur UI di JavaFX.
- **Relasi**: Setiap file FXML (.fxml) memiliki hubungan satu-ke-satu dengan sebuah kelas Controller. Sesuai konvensi yang dijelaskan dalam diagram, LoginController akan mengontrol login.fxml, DashboardController mengontrol dashboard.fxml, dan seterusnya.

3.2.3 Layer Controller

Lapisan ini bertindak sebagai jembatan antara View dan Model/Service.

- **Tujuan**: Menerima input dari pengguna (misalnya, klik tombol dari View), memprosesnya, dan kemudian memanggil logika bisnis yang sesuai di Service Layer. Setelah logika bisnis selesai, Controller akan memperbarui View untuk menampilkan hasilnya.
- **Kelas Utama**: LoginController, DashboardController, AdminController, LelangController, dll.

- **Dependensi**: Controller sangat bergantung pada Service untuk menjalankan tugas-tugas bisnis. Ia **tidak pernah** berinteraksi langsung dengan database. Controller juga menggunakan kelas dari Utility untuk fungsionalitas umum seperti navigasi dan menampilkan notifikasi.

3.2.4. Layer Service

Ini adalah lapisan krusial yang berisi logika bisnis aplikasi.

- **Tujuan**: Mengenkapsulasi semua aturan bisnis dan orkestrasi proses. Misalnya, logika untuk "memfinalisasi lelang" yang melibatkan penentuan pemenang, pengembalian kredit, dan pengiriman notifikasi, semuanya diatur di dalam LelangService.
- **Kelas Utama**: UserService, LelangService, KreditService, dll. Setiap Service fokus pada satu domain bisnis.
- **Dependensi**: Service bergantung pada Data Access Layer (DatabaseConnection) untuk mengambil dan menyimpan data. Ia juga berinteraksi dengan berbagai kelas Model untuk melakukan operasinya.

3.2.5. Layer Data Access

Lapisan ini bertanggung jawab atas semua komunikasi dengan sumber data (database).

- **Tujuan**: Mengabstraksi detail teknis dari operasi database (CRUD - Create, Read, Update, Delete).
- **Kelas Utama**: **DatabaseConnection**.
- **Pola Desain**: Kelas ini diimplementasikan menggunakan **Singleton Pattern**, yang memastikan bahwa hanya ada satu instance koneksi database yang aktif di seluruh aplikasi. Ini menghemat sumber daya dan mencegah masalah koneksi.

3.2.6. Layer Utility

Lapisan ini berisi kelas-kelas bantuan yang menyediakan fungsionalitas umum dan dapat digunakan kembali di seluruh aplikasi.

- **Tujuan**: Menyediakan fungsi-fungsi umum untuk menghindari duplikasi kode.
- **Kelas Utama**:
    - **SceneManager**: Mengelola semua navigasi dan transisi antar halaman (scene) di JavaFX.
    - **UIUtils**: Menyediakan fungsi-fungsi umum untuk UI, seperti menampilkan pesan peringatan (showAlert), konfirmasi, dan validasi input.

3.3 Alur Interaksi & Relasi

Diagram ini menunjukkan alur kerja yang jelas dan terstruktur:

1. Pengguna berinteraksi dengan **View** (misalnya, menekan tombol "Login").
2. **Controller** (LoginController) yang terhubung dengan View tersebut akan menangani aksi pengguna.
3. Controller memanggil method yang sesuai dari **Service** (UserService.authenticate()).
4. **Service** menjalankan logika bisnisnya. Untuk otentikasi, ia perlu data dari database.
5. **Service** menggunakan **Data Access** (DatabaseConnection.getInstance()) untuk berkomunikasi dengan database.
6. Setelah data didapat dan divalidasi, Service mengembalikan hasilnya (objek User atau null) ke Controller.
7. Controller kemudian memutuskan aksi selanjutnya: jika berhasil, ia akan menggunakan SceneManager dari **Utility** untuk pindah ke halaman dashboard; jika gagal, ia akan menggunakan UIUtils untuk menampilkan pesan error di **View**.

3.4. Pola Desain yang Digunakan

Beberapa pola desain utama yang diimplementasikan dalam sistem ini adalah:

- **Model-View-Controller (MVC)**: Pola arsitektur utama yang memisahkan data, tampilan, dan logika kontrol.
- **Layered Architecture**: Membagi sistem menjadi lapisan-lapisan independen (View, Controller, Service, Data Access) untuk meningkatkan modularitas.
- **Service Layer**: Mengisolasi logika bisnis dari Controller, membuatnya lebih mudah diuji dan dikelola.
- **Singleton Pattern**: Digunakan pada DatabaseConnection untuk memastikan hanya ada satu titik akses ke database.
- **Data Access Object (DAO)** (Tersirat): Lapisan Service dan Data Access secara bersama-sama membentuk pola DAO, di mana detail persistensi data disembunyikan dari seluruh aplikasi.

3.4. Entity Relationship Diagram

3.4.1. Penjelasan Hubungan Antar Entitas

Sistem StatCredit memiliki 11 entitas utama yang saling terhubung melalui berbagai jenis relasi. Hubungan antar entitas dalam sistem ini dapat dikategorikan berdasarkan kardinalitas sebagai berikut:

Hubungan One-to-One (1:1)

- ***user dan mahasiswa:*** Setiap pengguna yang berperan sebagai mahasiswa memiliki satu record di tabel mahasiswa. Relasi ini diimplementasikan melalui foreign key mahasiswa.id yang mereferensi user.id dengan constraint CASCADE.

Hubungan One-to-Many (1:N)

- ***kategori_kredit ke sub_kategori_kredit:*** Setiap kategori kredit dapat memiliki beberapa sub-kategori dengan nilai poin yang berbeda.
- ***mahasiswa ke pengajuan_kredit:*** Setiap mahasiswa dapat mengajukan beberapa kredit dari berbagai kategori dan sub-kategori.
- ***mahasiswa ke notifikasi:*** Setiap mahasiswa dapat menerima beberapa notifikasi terkait status pengajuan dan informasi sistem.

Hubungan Many-to-Many (N:M)

- ***mahasiswa dan lelang:*** Mahasiswa dapat mengikuti beberapa lelang penempatan kerja, dan setiap lelang dapat diikuti oleh beberapa mahasiswa. Hubungan ini diimplementasikan melalui tabel junction penawaran_lelang.
- ***mahasiswa dan barang_kopma:*** Mahasiswa dapat membeli beberapa jenis barang, dan setiap barang dapat dibeli oleh beberapa mahasiswa. Hubungan ini diimplementasikan melalui tabel junction transaksi_kopma.

Entitas Independen

- **events**: Entitas ini berdiri sendiri dan tidak memiliki hubungan foreign key dengan entitas lain, berfungsi untuk menyimpan informasi event kampus.

3.4.2. Diagram ERD

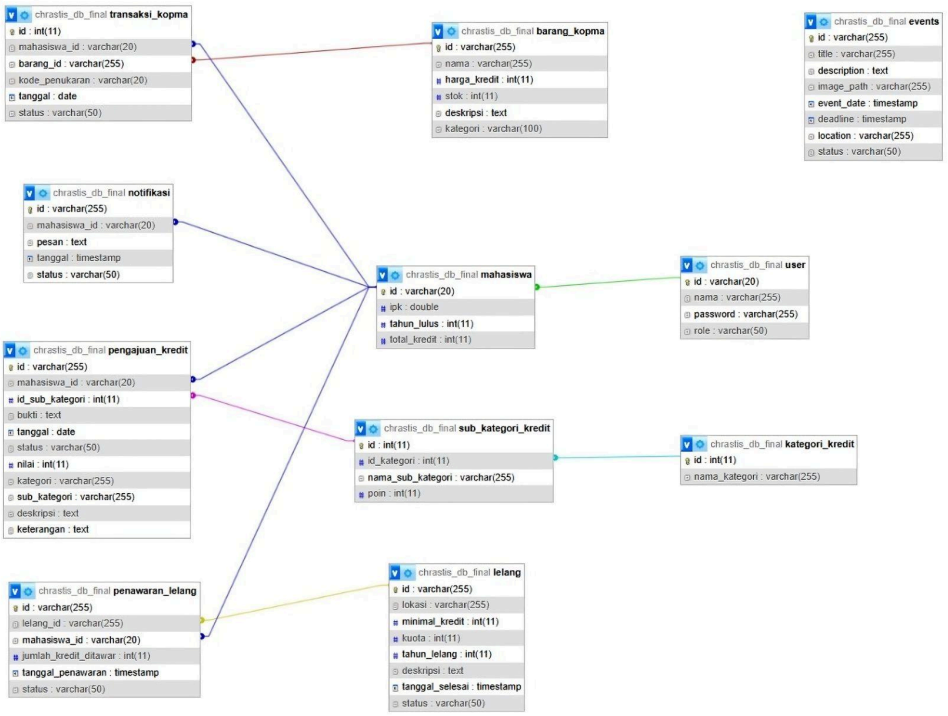


Diagram ERD di atas menggambarkan struktur database sistem StatCredit dengan 11 entitas yang saling terhubung. Entitas utama adalah MAHASISWA yang menjadi pusat dari sebagian besar relasi dalam sistem. Setiap entitas memiliki atribut yang ditandai dengan Primary Key (PK) dan Foreign Key (FK) sesuai dengan perannya dalam relasi database.

BAB IV
PENGEMBANGAN SISTEM

4.1. Konsep Program

Pengembangan StatCredit mengadopsi pola arsitektur **Model-View-Controller (MVC)** yang sangat cocok untuk aplikasi JavaFX:

- **Model**: Terdiri dari kelas-kelas Java (POJO - Plain Old Java Object) seperti Mahasiswa, BarangKopma, PengajuanKredit, serta kelas-kelas pada *Service Layer* yang bertanggung jawab untuk mengakses dan memanipulasi data di database.
- **View**: Didefinisikan dalam file FXML. FXML adalah bahasa markup berbasis XML yang digunakan untuk mendesain dan menyusun tata letak antarmuka pengguna, memisahkannya dari kode logika aplikasi.
- **Controller**: Kelas Java yang dihubungkan ke file FXML. Kelas ini berisi logika untuk menangani *event* dari pengguna (misalnya, klik tombol, input teks) dan bertindak sebagai jembatan antara View dan Model/Service.

4.2. Penjelasan Penggunaan dan Kode Program dengan Konsep *MVC*, *Design Principles*, dan *Design Patterns*

Fokus pengembangan pada tahap ini adalah implementasi GUI dan *event-handling*. Mekanisme ini adalah inti dari interaksi pengguna dengan aplikasi.

Contoh Kasus: Mahasiswa Menukar Credit dengan Barang

Dalam skenario ini, seorang mahasiswa ingin menukarkan poin kredit yang dimilikinya dengan barang yang tersedia di Koperasi Mahasiswa (KOPMA).

View

Tampilan untuk fitur ini didefinisikan dalam file Kopma.fxml. File ini berisi semua komponen visual seperti tabel untuk menampilkan barang, kolom-kolomnya, dan tombol aksi untuk setiap baris barang.

- **Pemisahan Tampilan (Separation of Concerns)**: FXML memungkinkan desainer UI untuk fokus pada tata letak tanpa harus menyentuh logika Java. fx:id digunakan untuk memberi nama unik pada setiap komponen agar dapat diakses oleh Controller, dan onAction digunakan untuk menautkan aksi tombol ke sebuah method di Controller.

Kode Contoh (Kopma.fxml - Potongan)

```
<TableView fx:id="barangTable" VBox.vgrow="ALWAYS">
  <columns>
          <TableColumn   fx:id="namaBarangColumn"   text="Nama   Barang"
```

```
prefWidth="250.0" />
    <TableColumn fx:id="hargaKreditColumn" text="Harga (Kredit)" />
    <TableColumn fx:id="stokColumn" text="Stok" />
    <TableColumn fx:id="aksiColumn" text="Aksi" prefWidth="120.0" />
  </columns>
</TableView>
```

Controller

Kelas KopmaController.java adalah otak di balik tampilan Kopma.fxml. Ia bertanggung jawab untuk memuat data barang, menampilkannya di tabel, dan menangani aksi "Tukar" dari pengguna.

- **Prinsip Single Responsibility**: Controller ini hanya bertanggung jawab untuk logika yang terkait dengan halaman KOPMA.
- **Pola Design Service Layer**: Controller tidak berinteraksi langsung dengan database. Sebaliknya, ia memanggil KopmaService yang berisi logika bisnis untuk proses penukaran barang. Ini membuat Controller lebih ramping dan logika bisnisnya dapat digunakan kembali.

Kode Contoh (KopmaController.java - Potongan)

```
public class KopmaController implements Initializable {
    @FXML private TableView<BarangKopma> barangTable;
    @FXML private TableColumn<BarangKopma, Void> aksiColumn;

    private Mahasiswa mahasiswa;
    private final KopmaService kopmaService = new KopmaService();

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        // ... kode inisialisasi lain ...
        setupAksiColumn(); // Menyiapkan tombol "Tukar" di dalam tabel
    }

    // Metode ini menyiapkan tombol "Tukar" untuk setiap baris di tabel
    private void setupAksiColumn() {
        aksiColumn.setCellFactory(param -> new TableCell<>() {
            private final Button tukarButton = new Button("🛒 Tukar");
            {
                    // Menetapkan aksi yang akan dijalankan saat tombol
diklik
                tukarButton.setOnAction(e -> {
```

```
                                                BarangKopma barang =
getTableView().getItems().get(getIndex());
                        handleTukarBarang(barang); // Memanggil handler
utama
                });
            }

            @Override
            protected void updateItem(Void item, boolean empty) {
                super.updateItem(item, empty);
                if (empty) {
                    setGraphic(null);
                } else {
                            // Logika untuk menonaktifkan tombol jika
kredit/stok tidak cukup
                                                BarangKopma barang =
getTableView().getItems().get(getIndex());
                        tukarButton.setDisable(barang.getStok() <= 0 ||
mahasiswa.getTotalKredit() < barang.getHargaKredit());
                    setGraphic(tukarButton);
                }
            }
        });
    }

    // Handler utama untuk proses penukaran
    private void handleTukarBarang(BarangKopma barang) {
        // 1. Validasi awal di sisi Controller
        if (mahasiswa.getTotalKredit() < barang.getHargaKredit()) {
               UIUtils.showAlert(Alert.AlertType.WARNING, "Kredit Tidak
Cukup", ...);
            return;
        }

        // 2. Konfirmasi dari pengguna
                    UIUtils.showConfirmation("Konfirmasi Penukaran",
...).ifPresent(response -> {
            if (response == ButtonType.OK) {
                // 3. Memanggil Service untuk menjalankan logika bisnis
                                                boolean success =
kopmaService.tukarBarang(mahasiswa.getId(), barang.getId());

                  // 4. Memberikan umpan balik ke View berdasarkan hasil
```

```
dari Service
                if (success) {
                        UIUtils.showAlert(Alert.AlertType.INFORMATION,
"Berhasil", ...);
                         initializeData(); // Memuat ulang data untuk
me-refresh tampilan
                } else {
                      UIUtils.showAlert(Alert.AlertType.ERROR, "Gagal",
...);
                }
            }
        });
    }

    public void initializeData() {
          // ... (memuat data mahasiswa, barang, dan riwayat transaksi)
...
    }
}
```

4.3. Penjelasan Penggunaan dan Kode Program dengan Konsep *Networking*

Meskipun StatCredit adalah aplikasi desktop, konsep *networking* (jaringan) tetap berperan penting dalam komunikasinya dengan database server. Interaksi ini terjadi melalui **JDBC (Java Database Connectivity)**, yang memungkinkan aplikasi Java untuk mengirimkan perintah SQL dan menerima data dari berbagai sistem database melalui jaringan.

a. Implementasi Koneksi Database (JDBC)

   Koneksi ke database dienkapsulasi dalam sebuah kelas khusus, DatabaseConnection, yang menerapkan **Singleton Design Pattern**. Pola ini memastikan bahwa hanya ada satu objek koneksi yang dibuat dan digunakan di seluruh aplikasi, sehingga menghemat sumber daya dan menjaga konsistensi.

Kode Contoh (DatabaseConnection.java)

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;



public class DatabaseConnection {
```

```
    // URL koneksi JDBC. "localhost:3306" adalah alamat server database di jaringan.
                    private     static     final     String     URL     = "jdbc:mysql://localhost:3306/statcredit_db";
    private static final String USER = "root";
      private static final String PASSWORD = ""; // Asumsi password kosong


    private static Connection connection;


    // Private constructor untuk mencegah instansiasi dari luar (ciri khas Singleton)
    private DatabaseConnection() {}


    /**
     * Metode static untuk mendapatkan satu-satunya instance koneksi.
     * Jika belum ada, koneksi akan dibuat.
     * @return Connection object
     */
    public static Connection getInstance() {
        if (connection == null) {
            try {
                  // Proses koneksi jaringan ke server database terjadi di sini
                    connection = DriverManager.getConnection(URL, USER, PASSWORD);
                       System.out.println("Koneksi jaringan ke database berhasil.");
```

```
            } catch (SQLException e) {
                System.err.println("Gagal terkoneksi ke database: " +
e.getMessage());
                // Handle error, misalnya dengan menampilkan alert
            }
        }
        return connection;
    }
}
```

b. Penggunaan Koneksi dalam Service Layer

Kelas-kelas *service* menggunakan koneksi yang disediakan oleh DatabaseConnection untuk menjalankan query SQL. Setiap query yang dieksekusi adalah bentuk komunikasi jaringan, di mana aplikasi (klien) mengirim permintaan (SQL) ke server database dan menunggu respons.

Kode Contoh (KopmaService.java - Potongan)

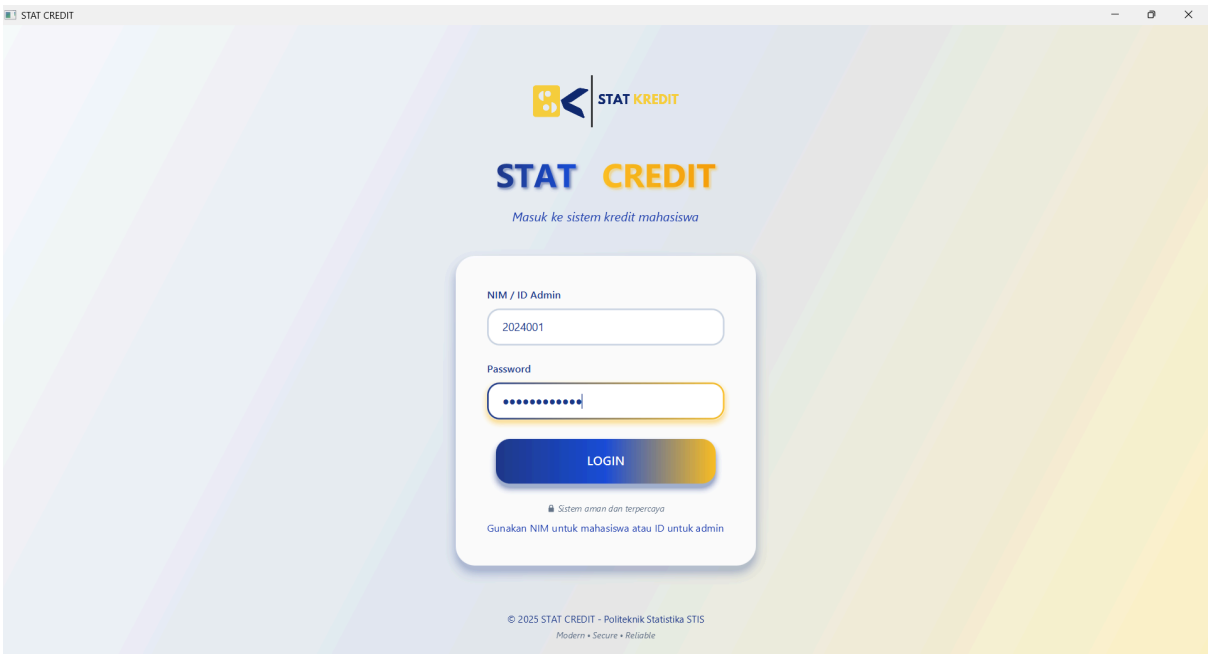
```
import java.sql.Connection;
import java.sql.PreparedStatement;

public class KopmaService {

    public boolean tukarBarang(String mahasiswaId, String barangId) {
                String sqlTransaksi = "INSERT INTO transaksi_kopma
(mahasiswa_id, barang_id, ...) VALUES (?, ?, ...)";
          String sqlUpdateKredit = "UPDATE mahasiswa SET total_kredit =
total_kredit - ? WHERE id = ?";
         String sqlUpdateStok = "UPDATE barang_kopma SET stok = stok - 1
WHERE id = ?";
```

```
        // Mendapatkan koneksi jaringan dari Singleton
        Connection conn = DatabaseConnection.getInstance();

        try {
            // Memulai transaksi
            conn.setAutoCommit(false);

                // 1. Eksekusi query INSERT (mengirim data melalui
jaringan)
                                    try (PreparedStatement ps =
conn.prepareStatement(sqlTransaksi)) {
                // ... set parameters ...
                ps.executeUpdate();
            }

              // 2. Eksekusi query UPDATE kredit (mengirim data melalui
jaringan)
                                    try (PreparedStatement ps =
conn.prepareStatement(sqlUpdateKredit)) {
                // ... set parameters ...
                ps.executeUpdate();
            }

               // 3. Eksekusi query UPDATE stok (mengirim data melalui
jaringan)
```

```
                                    try (PreparedStatement ps =
conn.prepareStatement(sqlUpdateStok)) {

                // ... set parameters ...

                ps.executeUpdate();

            }



            // Jika semua berhasil, commit transaksi

            conn.commit();

            return true;



        } catch (SQLException e) {

            // Jika ada error, rollback semua perubahan

                try { conn.rollback(); } catch (SQLException ex) { /*
handle ex */ }

            return false;

        } finally {

            // Kembalikan ke mode auto-commit

             try { conn.setAutoCommit(true); } catch (SQLException e) {
/* handle e */ }

        }

    }

}
```

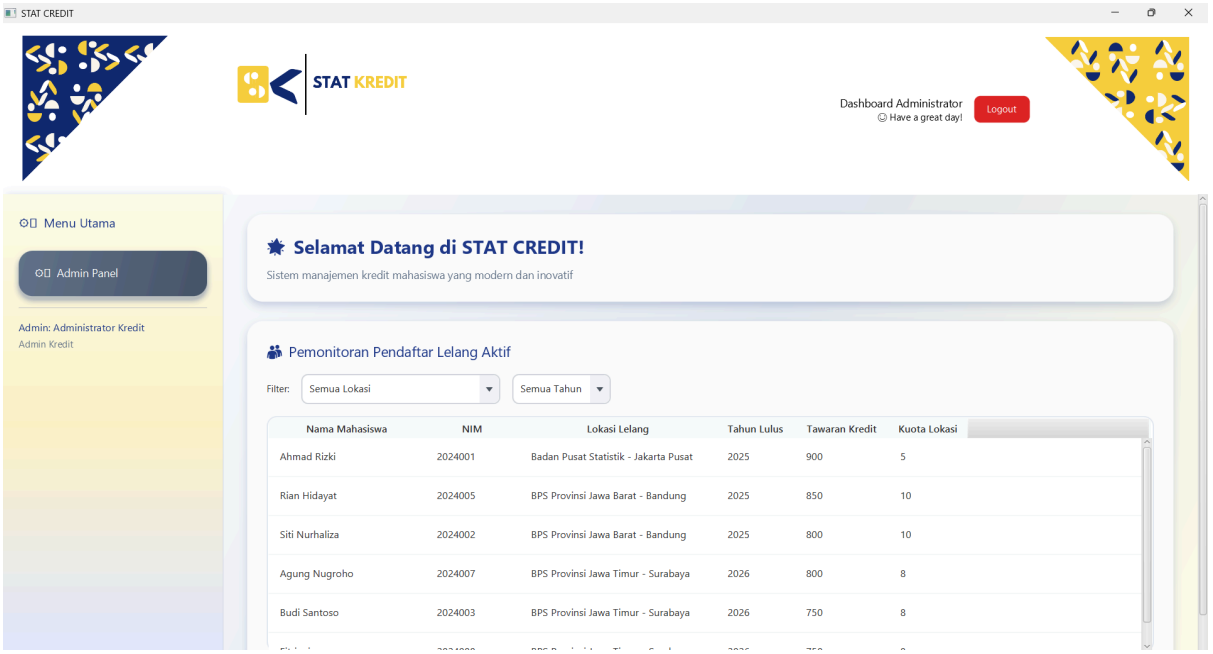
Dari contoh di atas, jelas bahwa setiap kali aplikasi perlu membaca atau mengubah data, ia melakukan operasi jaringan melalui JDBC ke server database.

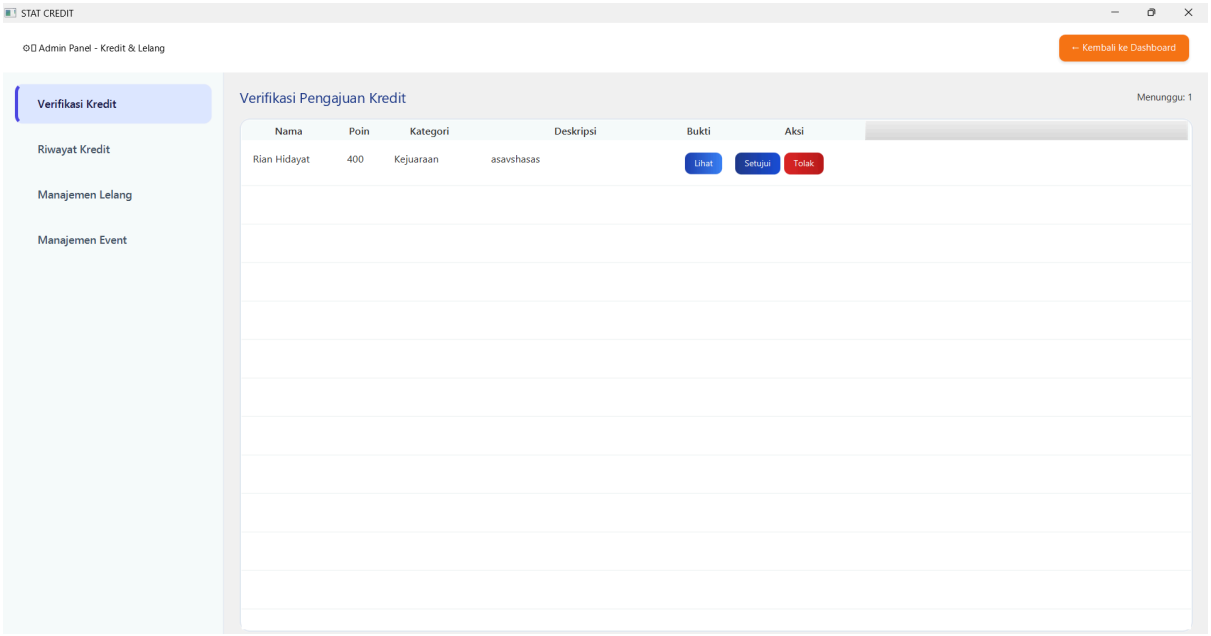
BAB V
IMPLEMENTASI DAN PANDUAN PENGGUNAAN



Gambar 1. Halaman *Login*

Deskripsi: Jendela awal aplikasi tempat pengguna (Mahasiswa atau Admin) memasukkan username dan password untuk mengakses sistem.

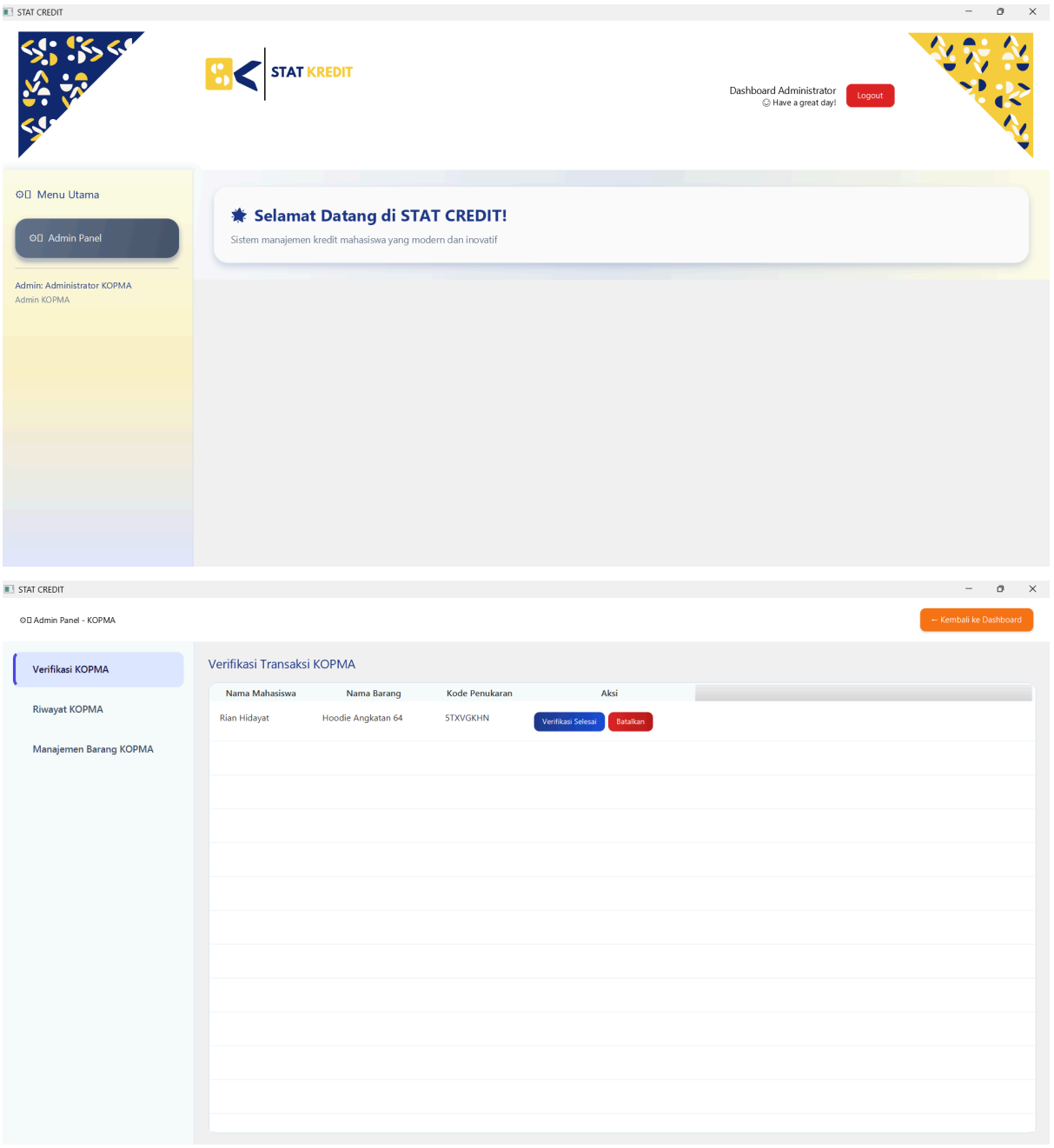




Gambar 2. Halaman *Dashboard* Admin Kredit
Deskripsi:
Halaman ini merupakan pusat kendali bagi Admin Kredit yang bertugas mengelola seluruh aktivitas terkait perolehan kredit mahasiswa dan lelang penempatan kerja.

Dashboard ini dirancang untuk memberikan kemudahan dalam pemantauan dan pengambilan keputusan melalui beberapa fitur utama:

- **Menu Navigasi Samping:** Terdapat panel navigasi di sisi kiri yang memungkinkan admin untuk mengakses modul-modul penting, seperti Verifikasi Kredit, Riwayat Kredit, Manajemen Lelang, dan Manajemen Event.
- **Pemantauan Lelang Aktif:** Bagian utama *dashboard* menampilkan tabel "Pemantauan Pendaftar Lelang Aktif" yang menyajikan data *real-time* mengenai mahasiswa yang berpartisipasi dalam lelang penempatan. Informasi yang ditampilkan mencakup Nama Mahasiswa, NIM, Lokasi Lelang yang diminati, Tahun Lulus, Jumlah Kredit yang ditawarkan, dan Kuota yang tersedia.
- **Fitur Filter:** Untuk memudahkan analisis, tersedia fungsi filter yang memungkinkan admin menyortir data pendaftar lelang berdasarkan "Semua Lokasi" dan "Semua Tahun".
- **Verifikasi Pengajuan Kredit:** Admin dapat masuk ke menu "Verifikasi Kredit" untuk meninjau dan memvalidasi setiap pengajuan kredit yang dikirimkan oleh mahasiswa. Di halaman ini, admin dapat melihat detail pengajuan seperti nama pengaju, jumlah poin, kategori, deskripsi, dan bukti pendukung. Admin memiliki tiga opsi aksi:
  Lihat untuk meninjau bukti, **Setuju** untuk menyetujui pengajuan, atau **Tolak** untuk menolaknya.



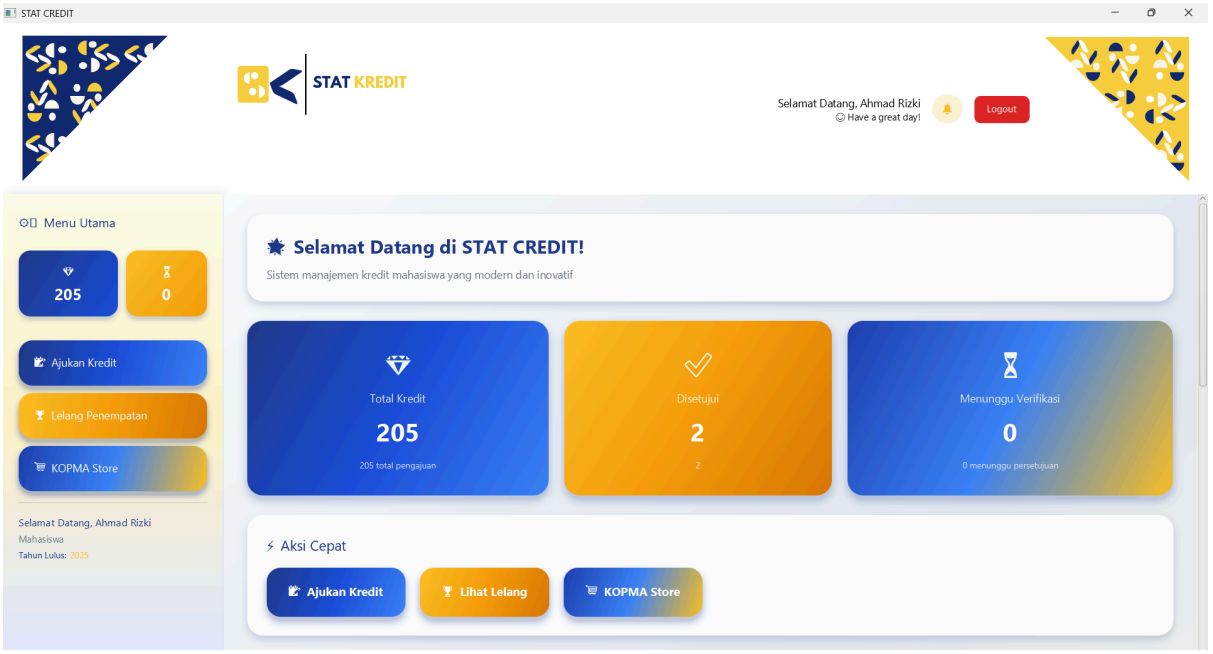
Gambar 3. Halaman *Dashboard* Admin Kopma
Deskripsi:
Halaman ini dirancang khusus untuk Admin Koperasi Mahasiswa (Kopma) yang berperan dalam mengelola proses penukaran (

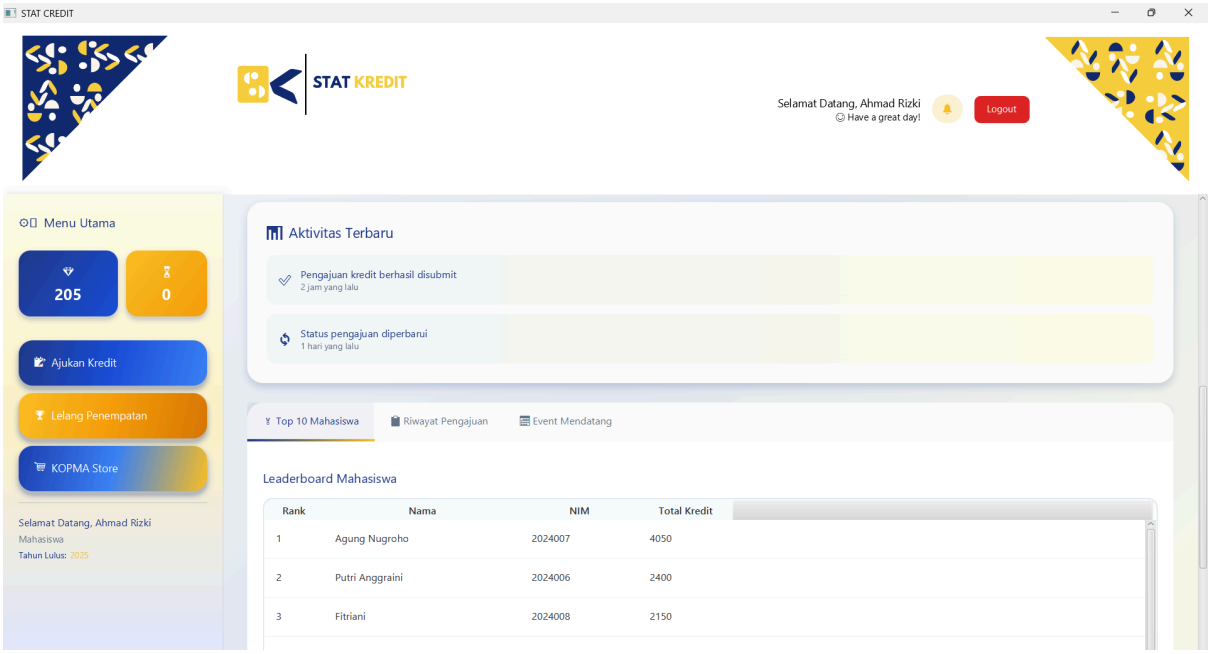
redeem) kredit mahasiswa dengan barang-barang yang tersedia di koperasi. Halaman ini memastikan bahwa setiap transaksi penukaran tercatat dan terverifikasi dengan benar. Fitur-fitur yang tersedia meliputi:

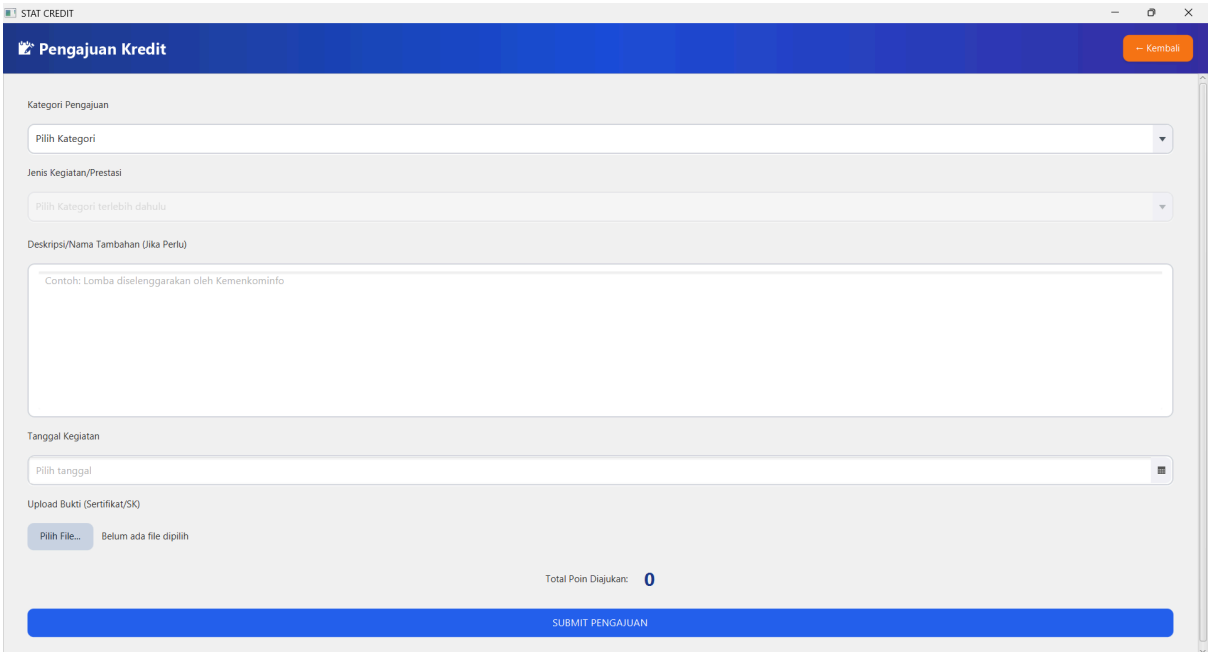
- **Menu Navigasi Samping:** Admin Kopma memiliki akses ke menu khusus yang

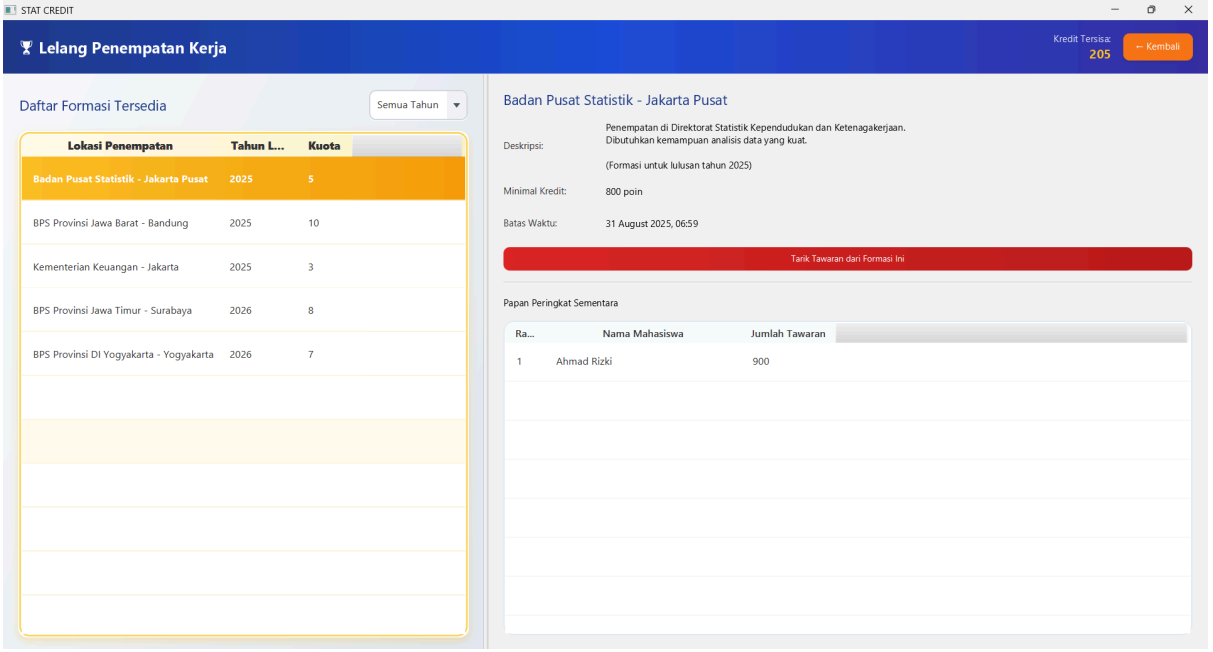
terdiri dari Verifikasi KOPMA, Riwayat KOPMA, dan Manajemen Barang KOPMA.

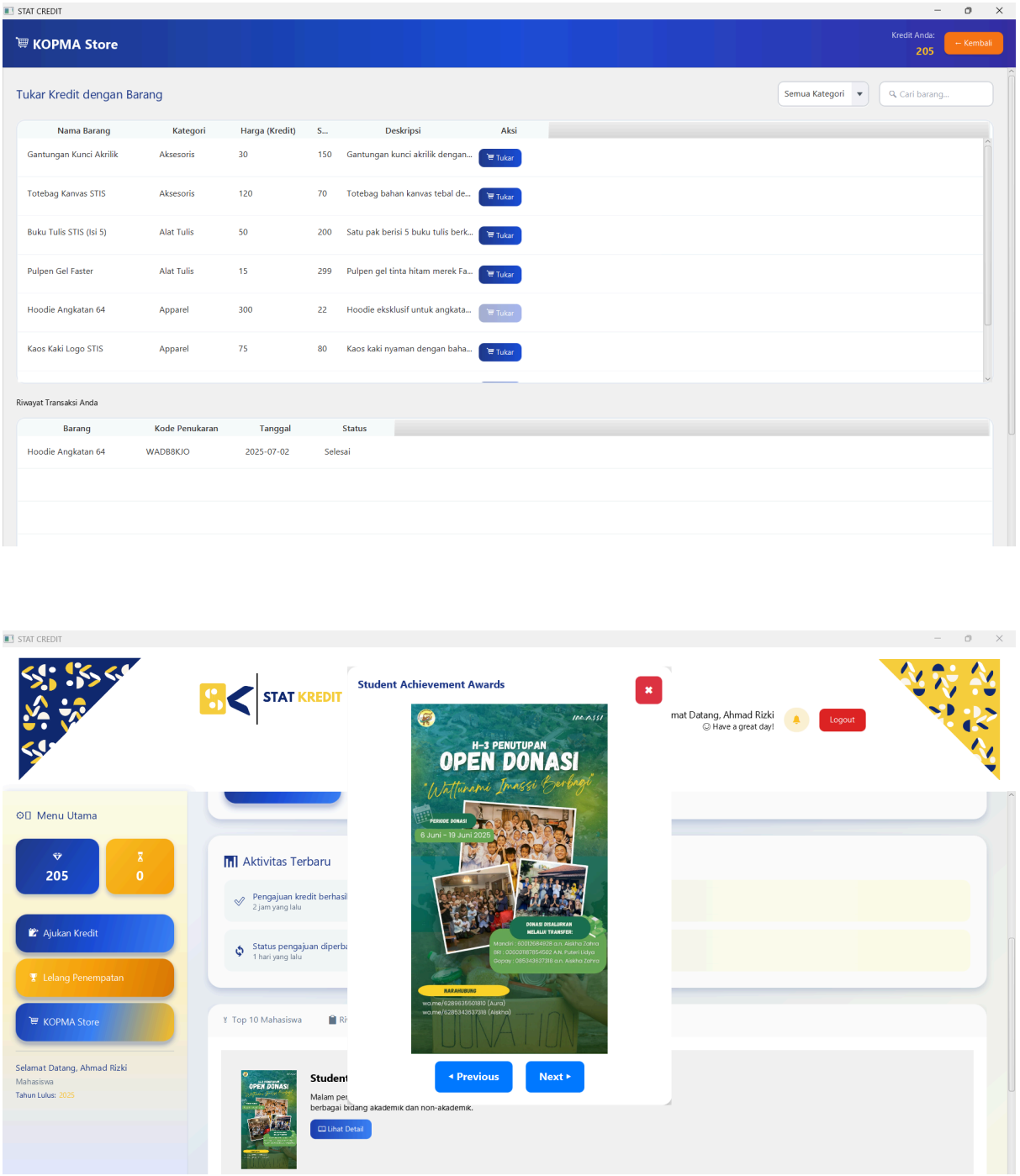
- **Verifikasi Transaksi:** Fitur inti dari *dashboard* ini adalah "Verifikasi Transaksi KOPMA". Halaman ini menampilkan daftar transaksi penukaran yang sedang berlangsung. Admin dapat melihat Nama Mahasiswa, Nama Barang yang ditukar, dan Kode Penukaran yang unik.
- **Aksi Verifikasi:** Untuk setiap transaksi, admin memiliki dua pilihan aksi: **Verifikasi Selesai** untuk mengonfirmasi bahwa barang telah diserahkan kepada mahasiswa dan transaksi selesai, atau **Batalkan** jika terjadi masalah dalam proses penukaran.
- **Manajemen Katalog Barang:** Melalui menu "Manajemen Barang KOPMA", admin dapat menambah, mengubah, atau menghapus item barang yang tersedia untuk penukaran, beserta harga dalam bentuk kredit dan jumlah stoknya.


STAT CREDIT
STAT KREDIT
Selamat Datang, Ahmad Rizki
Have a great day!
Logout
Menu Utama
205
0
Ajukan Kredit
Lelang Penempatan
KOPMA Store
Selamat Datang, Ahmad Rizki
Mahasiswa
Tahun Lulus: 2025
Selamat Datang di STAT CREDIT!
Sistem manajemen kredit mahasiswa yang modern dan inovatif
Total Kredit
205
205 total pengajuan
Disetujui
2
2
Menunggu Verifikasi
0
0 menunggu persetujuan
Aksi Cepat
Ajukan Kredit
Lihat Lelang
KOPMA Store


STAT CREDIT
STAT KREDIT
Selamat Datang, Ahmad Rizki
Have a great day!
Logout
Menu Utama
205
0
Ajukan Kredit
Lelang Penempatan
KOPMA Store
Selamat Datang, Ahmad Rizki
Mahasiswa
Tahun Lulus: 2025
Aktivitas Terbaru
Pengajuan kredit berhasil disubmit
2 jam yang lalu
Status pengajuan diperbarui
1 hari yang lalu
Top 10 Mahasiswa
Riwayat Pengajuan
Event Mendatang
Leaderboard Mahasiswa
Rank
Nama
NIM
Total Kredit
1
Agung Nugroho
2024007
4050
2
Putri Anggraini
2024006
2400
3
Fitriani
2024008
2150


STAT CREDIT
Pengajuan Kredit
← Kembali
Kategori Pengajuan
Pilih Kategori
Jenis Kegiatan/Prestasi
Pilih Kategori terlebih dahulu
Deskripsi/Nama Tambahan (Jika Perlu)
Contoh: Lomba diselenggarakan oleh Kemenkominfo
Tanggal Kegiatan
Pilih tanggal
Upload Bukti (Sertifikat/SK)
Pilih File...
Belum ada file dipilih
Total Poin Diajukan:
0
SUBMIT PENGAJUAN


STAT CREDIT
Lelang Penempatan Kerja
Kredit Tersisa:
205
← Kembali
Daftar Formasi Tersedia
Semua Tahun
Lokasi Penempatan
Tahun L...
Kuota
Badan Pusat Statistik - Jakarta Pusat
2025
5
BPS Provinsi Jawa Barat - Bandung
2025
10
Kementerian Keuangan - Jakarta
2025
3
BPS Provinsi Jawa Timur - Surabaya
2026
8
BPS Provinsi DI Yogyakarta - Yogyakarta
2026
7
Badan Pusat Statistik - Jakarta Pusat
Deskripsi:
Penempatan di Direktorat Statistik Kependudukan dan Ketenagakerjaan.
Dibutuhkan kemampuan analisis data yang kuat.
(Formasi untuk lulusan tahun 2025)
Minimal Kredit:
800 poin
Batas Waktu:
31 August 2025, 06:59
Tarik Tawaran dari Formasi Ini
Papan Peringkat Sementara
Ra...
Nama Mahasiswa
Jumlah Tawaran
1
Ahmad Rizki
900



Gambar 3. Halaman *Dashboard* Mahasiswa

Deskripsi:

Dashboard Mahasiswa adalah antarmuka utama bagi pengguna mahasiswa untuk berinteraksi dengan sistem StatCredit. Halaman ini didesain sebagai pusat informasi yang komprehensif dan interaktif, memungkinkan mahasiswa untuk memantau, mengajukan, dan memanfaatkan kredit mereka. Fitur-fitur utamanya adalah:

- **Ringkasan Poin:** Di bagian atas, terdapat kartu ringkasan (*summary cards*) yang menampilkan informasi krusial secara visual, yaitu **Total Kredit** yang dimiliki, jumlah pengajuan yang telah **Disetujui**, dan yang masih **Menunggu Verifikasi**.

- **Menu Aksi Utama:** Mahasiswa dapat dengan mudah mengakses fitur-fitur utama melalui menu navigasi di sisi kiri atau tombol "Aksi Cepat" di tengah halaman, yang mencakup:
    - **Ajukan Kredit:** Membawa mahasiswa ke formulir untuk mengajukan perolehan kredit baru dengan mengisi kategori, jenis kegiatan/prestasi, deskripsi, tanggal, dan mengunggah bukti pendukung.
    - **Lelang Penempatan:** Mengarahkan ke halaman di mana mahasiswa dapat melihat daftar formasi penempatan kerja yang tersedia, melihat detail persyaratan (seperti minimal kredit), dan menempatkan tawaran kredit untuk formasi yang diinginkan.
    - **KOPMA Store:** Membuka katalog barang dari koperasi mahasiswa yang dapat ditukar dengan kredit. Mahasiswa dapat melihat daftar barang, harga kredit, dan melakukan transaksi penukaran.
- **Aktivitas dan Informasi Tambahan:** *Dashboard* juga dilengkapi dengan fitur-fitur dinamis seperti:
    - **Aktivitas Terbaru:** Sebuah *feed* yang menampilkan notifikasi status pengajuan kredit terbaru.
    - **Leaderboard Mahasiswa:** Menampilkan peringkat mahasiswa dengan total kredit tertinggi, mendorong "gamifikasi" dan kompetisi yang sehat.
    - **Event Mendatang:** Informasi mengenai kegiatan atau acara kampus yang akan datang.

\

BAB VI
PENUTUP

6.1. Kesimpulan

Proyek pengembangan aplikasi "StatCredit" telah berhasil mencapai tujuannya untuk merancang dan membangun sebuah sistem desktop fungsional berbasis JavaFX. Aplikasi ini secara efektif menjawab kebutuhan akan adanya platform yang terpusat dan transparan untuk pencatatan, validasi, dan pemanfaatan kredit mahasiswa Politeknik Statistika STIS. Dengan mengadopsi arsitektur

Model-View-Controller (MVC), *Layered Architecture, Service Layer, Singleton Pattern,* dan *Data Access Object (DAO)*.

StatCredit menyediakan dua hak akses utama dengan fungsionalitas yang jelas: untuk

- **Mahasiswa**, aplikasi ini menjadi alat untuk memantau pencapaian, mengajukan kredit, serta memanfaatkannya dalam lelang penempatan kerja dan penukaran barang di koperasi. Sementara untuk
- **Admin** (Admin Kredit dan Admin Kopma), aplikasi ini berfungsi sebagai alat manajemen yang efisien untuk memvalidasi pengajuan, mengelola lelang, dan memverifikasi transaksi koperasi. Dengan demikian, StatCredit berhasil mentransformasi proses apresiasi keaktifan mahasiswa dari yang sebelumnya manual dan tersebar menjadi sebuah sistem yang terstruktur, terukur, dan memberikan insentif nyata.

6.2. Saran

Meskipun aplikasi StatCredit telah memenuhi ruang lingkup proyek yang ditetapkan, terdapat beberapa potensi pengembangan di masa depan untuk meningkatkan fungsionalitas dan pengalaman pengguna. Berikut adalah beberapa saran untuk pengembangan selanjutnya:

- **Pengembangan Versi Web dan *Mobile*:** Untuk meningkatkan aksesibilitas, disarankan untuk mengembangkan StatCredit dalam versi berbasis web atau aplikasi *mobile*. Hal ini akan memungkinkan pengguna untuk mengakses sistem dari berbagai perangkat tanpa perlu melakukan instalasi.
- **Peningkatan Fitur Notifikasi:** Sistem notifikasi dapat diperkaya dengan mengirimkan pemberitahuan secara *real-time* melalui *email* atau *push notification* (pada versi *mobile*) setiap kali ada pembaruan status pengajuan, informasi lelang baru, atau pengumuman penting lainnya.
- **Integrasi dengan Sistem Akademik:** Untuk meningkatkan efisiensi, aplikasi dapat diintegrasikan dengan sistem informasi akademik kampus. Hal ini memungkinkan

perolehan kredit dari kategori akademik (misalnya, IPK tinggi atau prestasi di kelas) dapat ditambahkan secara otomatis oleh sistem.

- **Modul Analitik dan Laporan:** Menambahkan modul *dashboard* analitik untuk admin yang menampilkan visualisasi data tren keaktifan mahasiswa, kategori kredit yang paling sering diajukan, atau barang kopma yang paling diminati. Fitur ini dapat memberikan wawasan berharga bagi institusi untuk pengambilan kebijakan.
- **Fitur *Gamifikasi* Lanjutan:** Selain *leaderboard*, fitur *gamifikasi* dapat diperluas dengan menambahkan sistem lencana (*badges*), pencapaian (*achievements*), atau level berdasarkan akumulasi kredit untuk lebih memotivasi mahasiswa agar terus aktif berkontribusi.

LAMPIRAN

Lampiran 1. Tautan *Source Code*
https://github.com/ahmadhn26/PBO_TA