PEMROGRAMAN BERBASIS OBJEK

PROJEK UAS : APLIKASI FinanSTIS

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



## Disusun Oleh:

Yedija Lewi Suryadi (222212921)

**PROGRAM STUDI D-IV KOMPUTASI STATISTIK POLITEKNIK STATISTIKA STIS**

**TAHUN AJARAN 2023/2024**

## 1. Pendahuluan

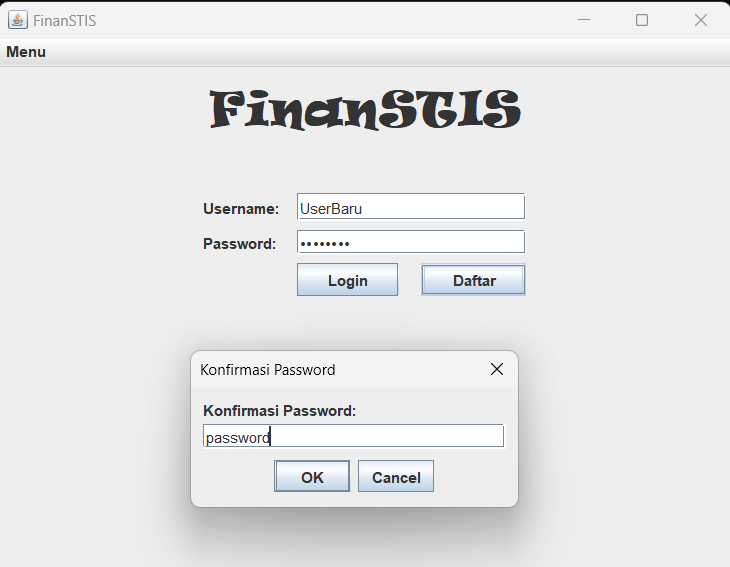
Aplikasi FinanSTIS adalah aplikasi manajemen keuangan pribadi yang memungkinkan pengguna untuk mencatat pemasukan, pengeluaran, transfer saldo antar kategori, dan melihat riwayat transaksi. Aplikasi ini dirancang dengan menggunakan konsep Model-View-Controller (MVC), prinsip desain SOLID, dan pola desain (design patterns) untuk memastikan arsitektur yang baik dan mudah dikembangkan.

**2. Tampilan Aplikasi**

**Halaman Login**

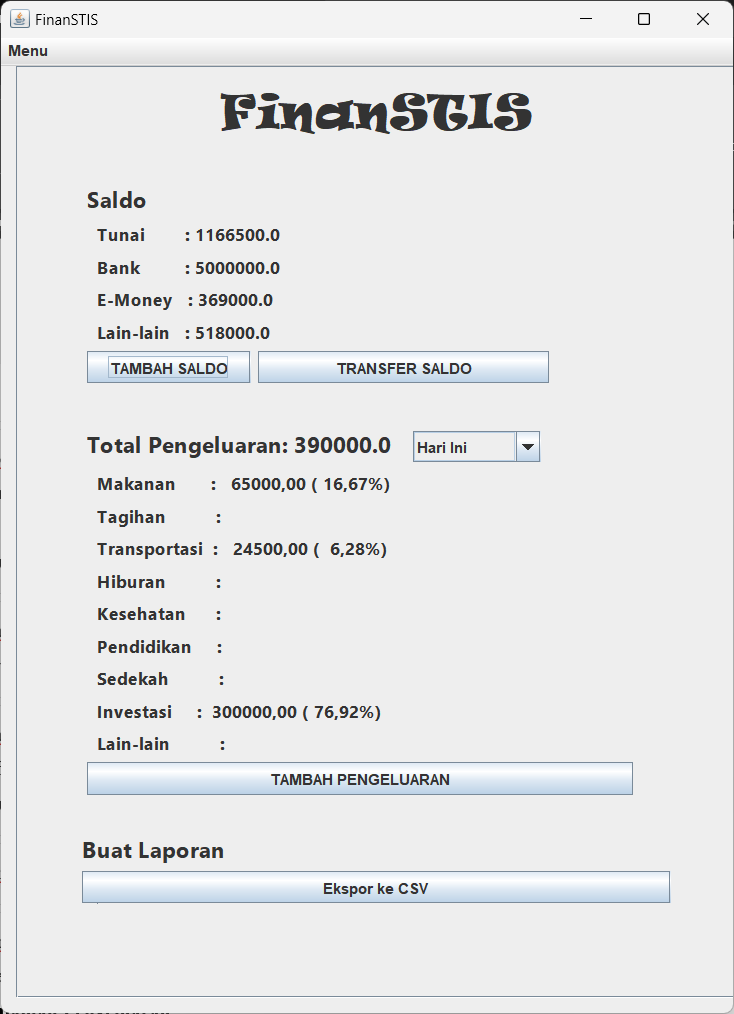
****

**Penjelasan Penggunaan:**  
Form ini digunakan oleh pengguna untuk masuk ke dalam aplikasi dengan mengisi username dan password yang telah terdaftar. Jika belum terdaftar, maka dapat menggunakan tombol daftar dan akan muncul pesan konfirmasi password. Jika berhasil, maka akan masuk ke dashboard.



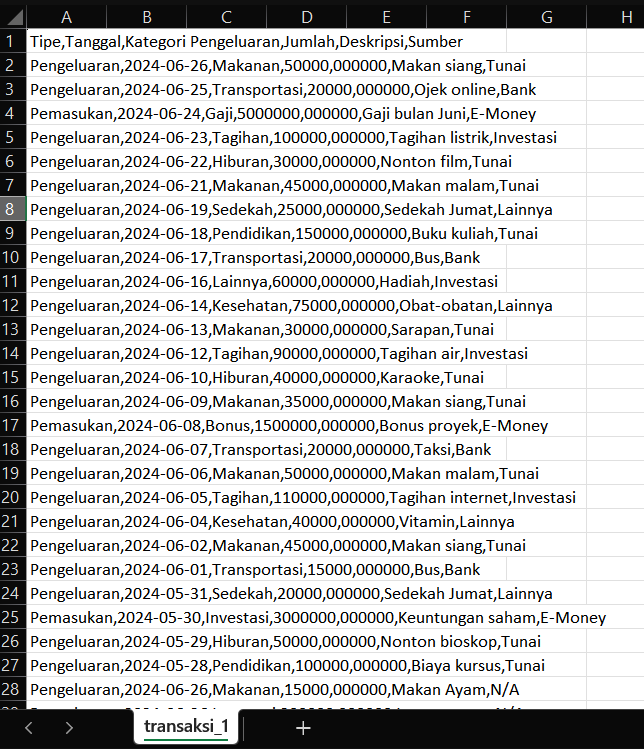
**Penjelasan Kode Program:**  
LoginPanel (View) berinteraksi dengan UserController (Controller) untuk memvalidasi pengguna terhadap data di users table (Model).

**Halaman Dashboard**

****

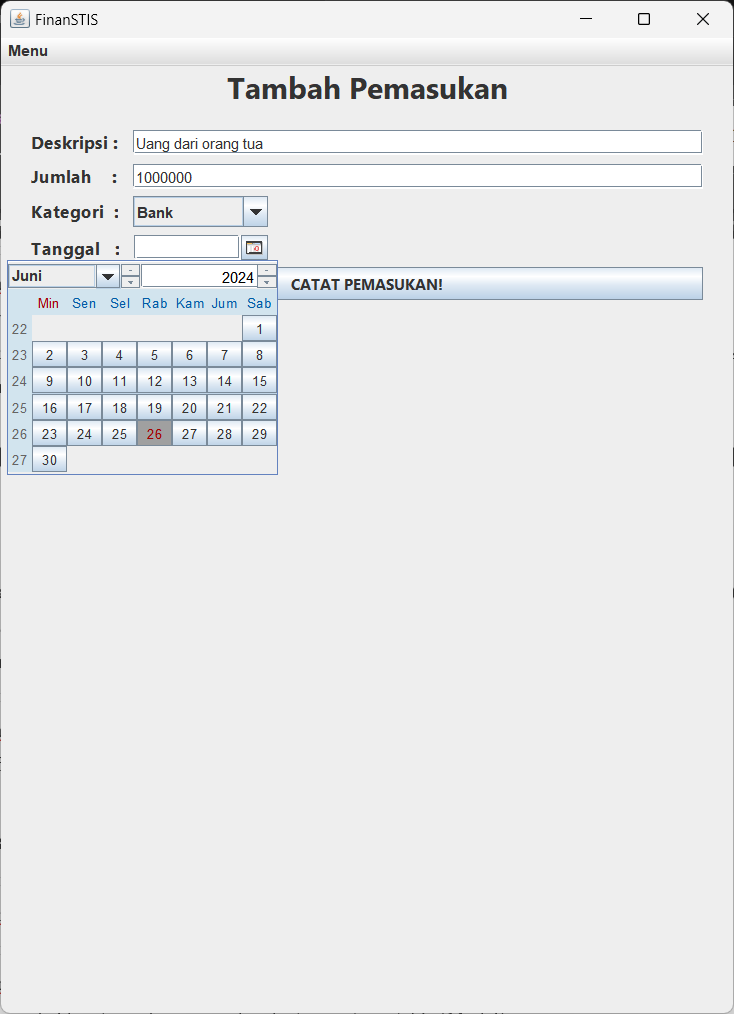
**Penjelasan Penggunaan:**  
Dashboard menampilkan ringkasan saldo, total pengeluaran, dan pemasukan dalam periode tertentu, serta fitur ekspor laporan dalam bentuk csv berisi daftar transaksi yang dilakukan. Terdapat Menu bar untuk navigasi ke panel lainnya. Ada juga tombol “Tambah Saldo” untuk ke panel Pemasukan yang berguna untuk menambah saldo, “Transfer Saldo” untuk ke panel Transfer yang berguna untuk memindahkan saldo antar sumber (contohnya dari bank ke tunai), “Tambah Pengeluaran” untuk navigasi ke panel Pengeluaran yang berguna untuk mencatat pengeluaran.

Fitur ekspor akan menghasilkan laporan dalam bentuk csv, contohnya adalah sebagai berikut.



**Penjelasan Kode Program:**  
DashboardPanel (View) memuat data dari ExpenseController (Controller) yang mendapatkan informasi dari balances dan transactions table (Model).

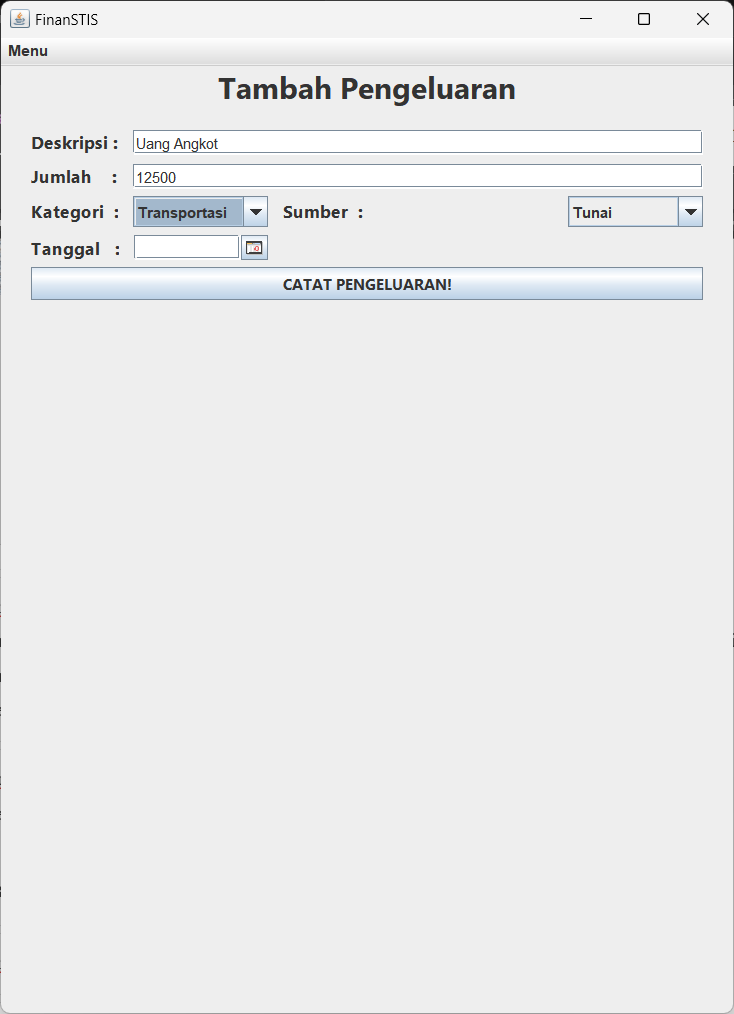
**Halaman Pemasukan**

****

**Penjelasan Penggunaan:**  
Form ini digunakan untuk mencatat pemasukan baru ke dalam sistem. Kolom deskripsi diisi dengan sumber pemasukan, jumlah diisi dengan jumlah uang yang akan ditambah, kategori diisi dengan rekening atau saldo tujuan yang akan ditambah, serta tanggal diisi dengan tanggal melakukan penambahan saldo.

**Penjelasan Kode Program:**  
PemasukanPanel (View) berinteraksi dengan ExpenseController (Controller) untuk menambahkan transaksi pemasukan ke transactions table (Model).

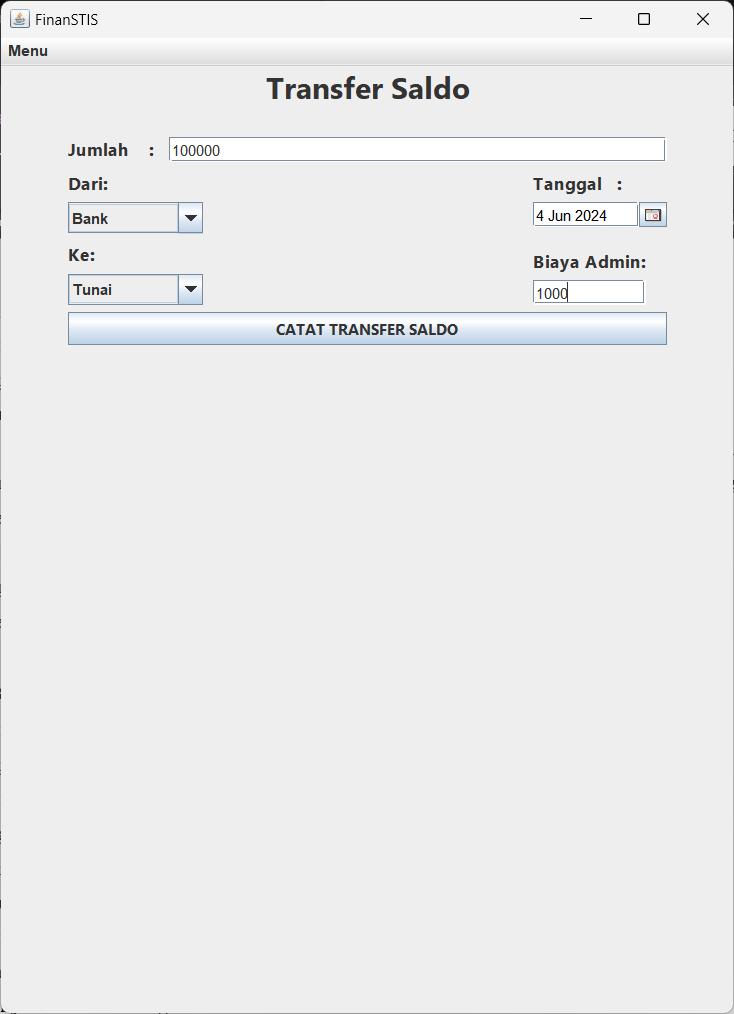
**Halaman Pengeluaran**

****

**Penjelasan Penggunaan:**  
Form ini digunakan untuk mencatat pengeluaran baru ke dalam sistem. Kolom deskripsi diisi dengan tujuan pengeluaran, jumlah diisi dengan jumlah uang yang akan dikeluarkan, kategori diisi dengan jenis pengeluaran yang ingin dilakukan (misalnya makanan, transportasi, dan lain-lain), sumber diisi dengan sumber saldo yang akan digunakan, serta tanggal diisi dengan tanggal melakukan pengeluaran.

**Penjelasan Kode Program:**  
PengeluaranPanel (View) berinteraksi dengan ExpenseController (Controller) untuk menambahkan transaksi pengeluaran ke transactions table (Model).

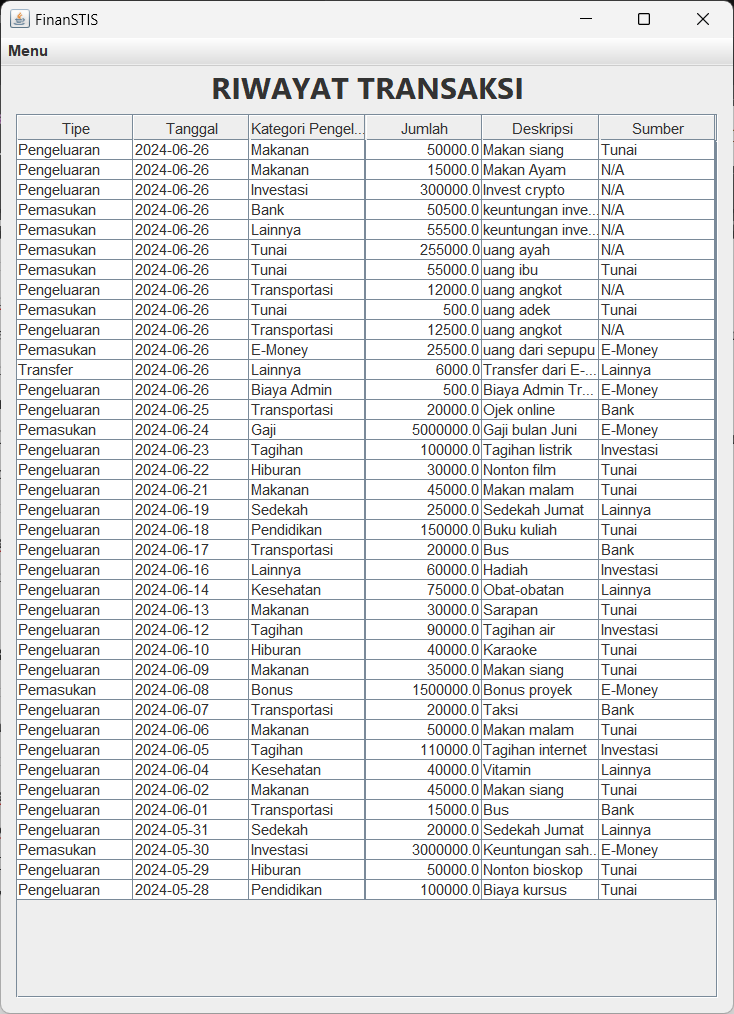
**Halaman Transfer Saldo**

****

**Penjelasan Penggunaan:**  
Form ini digunakan untuk mentransfer saldo antar kategori saldo milik pribadi (bukan kepada orang lain), termasuk biaya admin jika ada. Kolom jumlah diisi dengan jumlah uang yang akan ditransfer, kemudian kolom “dari” diisi dengan rekening atau saldo asal dan kolom “ke” diisi dengan rekening atau saldo tujuan, kolom tanggal diisi dengan tanggal melakukan transfer, kolom Biaya Admin diisi jika ada biaya admin yang dibebankan ketika melakukan transfer dan otomatis akan dimasukkan ke pengeluaran dengan sumber dana dari rekening asal.

**Penjelasan Kode Program:**  
TransferPanel (View) berinteraksi dengan ExpenseController (Controller) untuk mencatat transaksi transfer dan biaya admin ke transactions table (Model).

**Halaman Transaksi**

****

**Penjelasan Penggunaan:**  
Halaman ini menampilkan riwayat transaksi pengguna. Ditampilkan dalam bentuk tabel dengan kolom Tipe untuk tipe transaksi yang dilakukan, kolom Tanggal untuk tanggal transaksi dilakukan, kolom Kategori untuk kategori transaksi yang dilakukan, kolom jumlah untuk jumlah uang yang dilakukan saat transaksi, kolom deskripsi untuk mendeskripsikan transaksi yang dilakukan, dan kolom sumber untuk sumber saldo atau rekening yang digunakan untuk melakukan transaksi.

**Penjelasan Kode Program:**  
TransaksiPanel (View) mendapatkan dan menampilkan data transaksi dari ExpenseController (Controller).

**3. Penerapan MVC, Design Principles, dan Design Patterns**

**MVC (Model-View-Controller)**

* **Model:**  
  Model mencakup semua bagian yang terkait dengan data dan logika bisnis aplikasi. Di aplikasi FinanSTIS, ini termasuk kelas-kelas seperti Balance dan Transaction, serta interaksi dengan database melalui DatabaseHelper.
* **View:**  
  View bertanggung jawab untuk menampilkan data kepada pengguna dan menerima input dari pengguna. Di aplikasi FinanSTIS, ini termasuk berbagai panel seperti LoginPanel, DashboardPanel, PemasukanPanel, PengeluaranPanel, TransferPanel, dan TransaksiPanel.
* **Controller:**  
  Controller bertindak sebagai perantara antara Model dan View. Di aplikasi FinanSTIS, ini termasuk kelas ExpenseController dan UserController yang mengelola logika aplikasi dan memfasilitasi komunikasi antara Model dan View.

**Design Principles**

* **Single Responsibility Principle:**  
  Setiap kelas dalam aplikasi FinanSTIS memiliki satu tanggung jawab yang jelas. Misalnya, UserController hanya menangani logika pengguna, sementara ExpenseController menangani logika pengeluaran.
* **Open/Closed Principle:**  
  Kode dalam aplikasi FinanSTIS dirancang agar terbuka untuk ekstensi tetapi tertutup untuk modifikasi. Misalnya, metode dalam ExpenseController dapat diperluas untuk mendukung jenis transaksi baru tanpa mengubah kode yang ada.
* **Liskov Substitution Principle:**  
  Kelas turunan dalam aplikasi FinanSTIS dapat menggantikan kelas induk tanpa mengganggu fungsionalitas aplikasi. Misalnya, jika ada turunan dari Transaction, mereka dapat digunakan di tempat Transaction tanpa masalah.
* **Interface Segregation Principle:**  
  Antarmuka dalam aplikasi FinanSTIS dirancang agar spesifik dan tidak memaksa kelas untuk mengimplementasikan metode yang tidak mereka gunakan. Namun, aplikasi ini mungkin tidak menggunakan banyak antarmuka eksplisit karena sifat sederhana dari proyek.
* **Dependency Inversion Principle:**  
  Aplikasi FinanSTIS bergantung pada abstraksi daripada implementasi konkret. Misalnya, ExpenseController bergantung pada DatabaseOperations (antarmuka) daripada DatabaseOperationsImpl (implementasi konkret).

**Design Patterns**

* **Factory Pattern:**  
  MenuFactory digunakan untuk membuat menu bar aplikasi dengan cara yang terorganisir dan dapat diperluas.

# LAMPIRAN

Link Video Demo Aplikasi: <https://drive.google.com/drive/folders/1rrM2Lf_19zQ8wYaHGwYYh3rWHrTlknGv?usp=drive_link>

Link Github: <https://github.com/DijasanLewis/FinanSTIS.git>