

**UJIAN AKHIR SEMESTER**  
**PEMROGRAMAN BERORIENTASI OBJEK**  
**“SISTEM MANAJEMEN PARKIR POLITEKNIK STATISTIKA STIS”**



**Dosen Pengampu:**

Ibnu Santoso

**Disusun Oleh:**

Dwinanda Muhammad Keyzha (222212576)

**PROGRAM D-IV KOMPUTASI STATISTIK**

**POLITEKNIK STATISTIKA STIS**

**JL. OTTO ISKANDARDINATA, NO 64.C, JAKARTA TIMUR**

**2023/2024**

## KATA PENGANTAR

Puji Syukur kehadiran Tuhan Yang Maha Esa, atas berkat dan rahmat-Nya, penulis masih diberi kesehatan dan kesempatan dalam menyelesaikan tugas akhir semester dalam bentuk pembuatan aplikasi desktop sederhana dengan *software* yang berjudul “Sistem Manajemen Parkir Politeknik Statistika STIS” dengan tepat waktu.

Laporan ini disusun penulis dengan maksimal dan mendapatkan bantuan dari berbagai pihak sehingga dapat memperlancar pembuatan laporan aplikasi ini dalam rangka memenuhi ujian akhir semester mata kuliah pemrograman berorientasi objek. Penulis mengucapkan terima kasih kepada Bapak Ibnu Santoso selaku dosen pengampu mata kuliah pemrograman berorientasi objek kelas 2KS2 yang telah memberikan ilmu pengetahuan selama empat belas pertemuan baik melalui teori dan praktikum sehingga penulis mampu menyelesaikan pembuatan aplikasi ini untuk penilaian Ujian Akhir Semester (UAS).

Penulis menyadari makalah ini masih jauh dari sempurna. Oleh sebab itu kritik dan saran membangun sangat diperlukan demi kesempurnaan makalah ini. Penulis menyadari sepenuhnya bahwa masih ada kekurangan baik dari susunan kalimat dalam laporan dan fungsi dari aplikasi yang dibuat. Oleh karena itu, penulis sangat menerima saran dan kritik membangun dari pengguna agar penulis dapat menyempurnakan aplikasi ini. Akhir kata, penulis berharap semoga aplikasi ini dapat memberikan dampak positif bagi Politeknik Statistika STIS.

Jakarta, 26 Juni 2024

Dwinanda Muhammad Keyzha

# DAFTAR ISI

<b>UJIAN AKHIR SEMESTER PEMROGRAMAN BERORIENTASI OBJEK.....</b>	<b>i</b>
<b>KATA PENGANTAR.....</b>	<b>ii</b>
<b>BAB 1 PENDAHULUAN .....</b>	<b>1</b>
<b>1.1. Latar Belakang.....</b>	<b>1</b>
<b>1.2. Rumusan Masalah .....</b>	<b>2</b>
<b>1.3. Tujuan.....</b>	<b>2</b>
<b>BAB 2 PEMBAHASAN .....</b>	<b>3</b>
<b>2.1. Rancangan Class Diagram .....</b>	<b>3</b>
<b>2.2. Design Pattern .....</b>	<b>4</b>
<b>2.2.1. Singleton Pattern.....</b>	<b>4</b>
<b>2.2.2. Observer Pattern.....</b>	<b>4</b>
<b>2.3. Design Principle .....</b>	<b>5</b>
<b>2.4. Model View Controller (MVC) .....</b>	<b>6</b>
<b>BAB 3 ISI DAN DOKUMENTASI.....</b>	<b>8</b>
<b>3.1. Hasil Pengujian .....</b>	<b>8</b>
<b>3.2. Panduan Penggunaan dan Penjelasan Program .....</b>	<b>15</b>
<b>LINK VIDEO .....</b>	<b>26</b>
<b>LINK GIT REPOSITORY .....</b>	<b>26</b>

# **BAB 1**

## **PENDAHULUAN**

### **1.1. Latar Belakang**

Politeknik Statistika STIS, yang terletak di Jl. Otto Iskandardinata, No 64.C, Jakarta Timur, merupakan sebuah institusi pendidikan yang terus berkembang dengan jumlah mahasiswa, dosen, dan staff administratif yang semakin meningkat. Sebagai bagian dari infrastruktur kampus, Politeknik Statistika STIS menyediakan fasilitas parkir yang terletak di basement dan halaman depan kampus.

Hingga saat ini, pengelolaan lahan parkir di kampus ini masih dilakukan secara manual oleh satpam. Metode ini dapat menimbulkan berbagai tantangan, seperti kesulitan dalam manajemen penggunaan parkir secara efisien, serta pengumpulan data yang akurat dan teratur.

Dalam rangka meningkatkan efisiensi dan efektivitas manajemen parkir, serta menghadirkan pengalaman pengguna yang lebih baik bagi mahasiswa, dosen, dan staff, diperlukan implementasi sistem manajemen parkir yang modern dan terintegrasi di Politeknik Statistika STIS. Sistem ini dirancang untuk menggantikan pendekatan manual saat ini dengan solusi yang lebih otomatis, yang mampu memberikan keuntungan dalam penggunaan data untuk perencanaan dan pengambilan keputusan yang lebih baik. Dengan mengimplementasikan sistem manajemen parkir yang canggih dan terintegrasi, Politeknik Statistika STIS diharapkan dapat memajukan infrastruktur kampusnya dan memberikan standar pelayanan yang tinggi bagi seluruh komunitas akademiknya.

## **1.2. Rumusan Masalah**

Dalam konteks pengelolaan parkir di Politeknik Statistika STIS, berikut adalah tiga poin rumusan masalah yang perlu diselesaikan:

1. Bagaimana cara mengatasi tantangan dalam pengelolaan parkir secara manual yang saat ini dihadapi oleh Politeknik Statistika STIS?
2. Bagaimana cara mengatasi penghambat dalam pengalaman pengguna parkir (mahasiswa, dosen, dan staff) di kampus saat ini tersebut?
3. Bagaimana integrasi sistem keamanan parkir dengan teknologi yang ada dapat ditingkatkan untuk mengurangi risiko kehilangan kendaraan di area parkir kampus?

## **1.3. Tujuan**

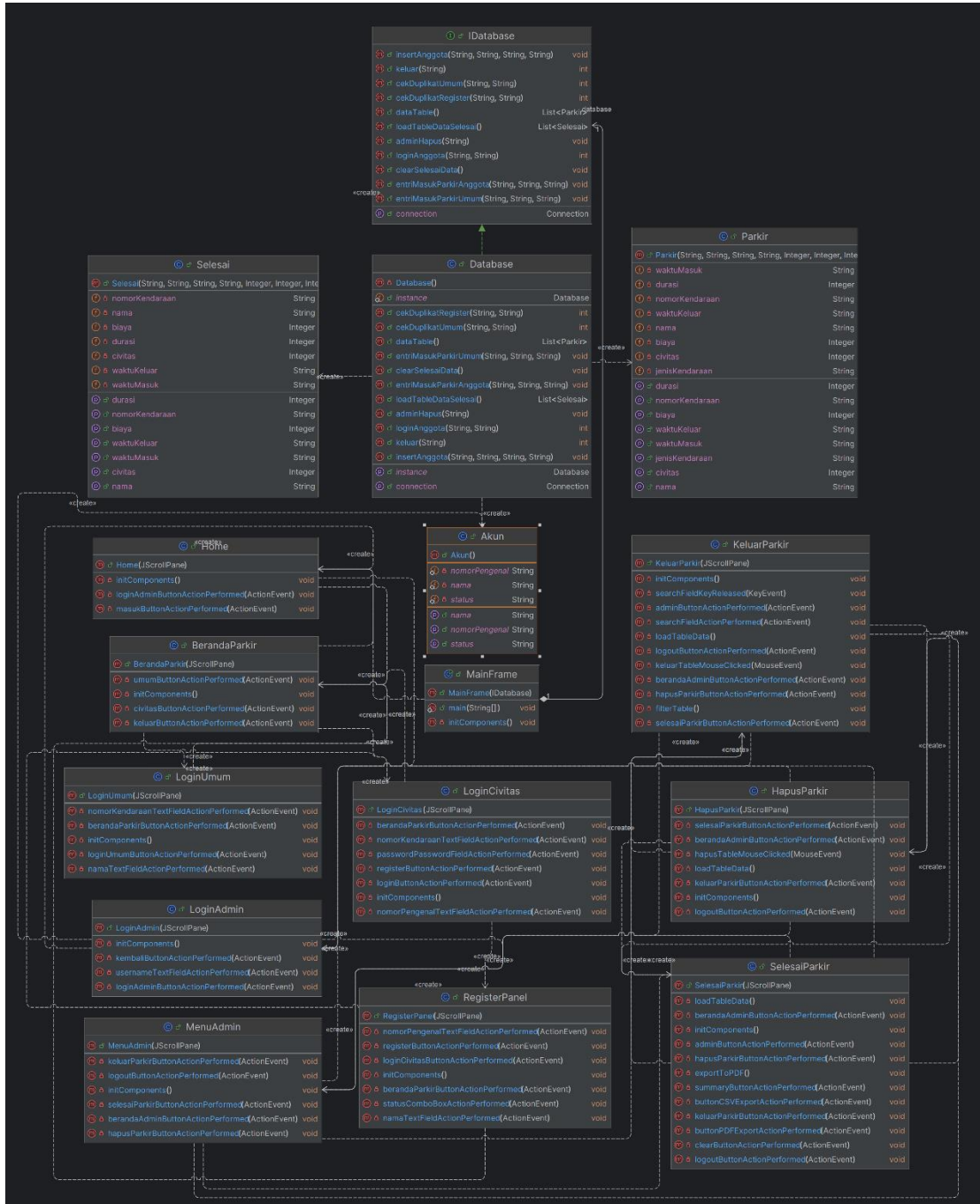
Tujuan dari penelitian ini adalah untuk:

1. Mengembangkan solusi yang menggunakan teknologi modern untuk optimalisasi penggunaan fasilitas parkir di Politeknik Statistika STIS.
2. Meningkatkan pengalaman pengguna parkir dengan menyediakan aksesibilitas yang lebih baik dan proses parkir kendaraan yang lebih nyaman di kampus dengan aplikasi.
3. Memperkuat keamanan parkir dengan mengintegrasikan sistem keamanan yang efektif, seperti pengawasan kamera dan teknologi identifikasi kendaraan, guna mengurangi kemungkinan terjadinya kehilangan kendaraan di area parkir kampus

## BAB 2

## PEMBAHASAN

### 2.1. Rancangan Class Diagram



## **2.2. Design Pattern**

### **2.2.1. Singleton Pattern**

Singleton pattern adalah salah satu design pattern yang memastikan bahwa kelas hanya memiliki satu instansi tunggal (instance) di dalam program, dengan memberikan akses secara global terhadap instance tersebut. Singleton pattern memiliki tujuan untuk membatasi pemberian akses yaitu hanya ke satu objek dimana pembuatan objek menjadi satu instansi saja. Singleton pattern diimplementasikan dengan menggunakan kelas statis, dimana method akan memeriksa apakah instansi sudah ada sebelumnya, jika belum ada maka instansi akan dibuat. Singleton pattern banyak digunakan pada koneksi database yang memastikan bahwa hanya ada satu koneksi yang terhubung ke database dan digunakan pada seluruh package atau aplikasi. Dalam pembuatan aplikasi ini, Singleton pattern diterapkan dengan mengimplementasikan Interface Serializable pada class Database. Singleton Design Pattern dapat digunakan untuk melakukan koneksi ke database, logging, dan seterusnya. Penerapan Singleton Pattern dalam aplikasi ini juga dapat terlihat dari beberapa kelas lain yang melakukan koneksi ke database dan memastikan hanya ada satu koneksi yang digunakan secara global, seperti LoginAdmin dan MenuAdmin.

Tanda-tandanya adanya singleton design pattern:

1. Terdapat default constructor secara private  
Memastikan bahwa tidak ada kelas lain yang dapat membuat instansi dari kelas tersebut. Dalam diagram, kelas Database memiliki constructor private Database().
2. Terdapat method statis yang bertindak secara constructor  
Menyediakan titik akses global dan kelas Database memiliki metode statis yang disinkronkan getInstance().

### **2.2.2. Observer Pattern**

Observer adalah behavioural design pattern yang menyediakan mekanisme berlangganan (subscription) bagi objek-objek yang ingin mengetahui perubahan yang terjadi pada suatu objek yang diamati. Pola ini digunakan dalam konteks listener pada komponen Java Swing.

Listener untuk Event: Implementasi action listener pada komponen GUI. Dalam diagram kelas, dapat dilihat pada metode yang ada di beberapa kelas seperti Home, BerandaParkir, LoginAdmin, dan lainnya, yang memiliki metode `actionPerformed(ActionEvent e)` menunjukkan penerapan Observer Pattern melalui mekanisme listener. ActionListener ini memberikan mekanisme untuk mendengar (mengamati) aksi tersebut dan meresponsnya sesuai. Method ini adalah tempat di mana logika aksi dituliskan. Ketika pengguna berinteraksi dengan GUI (misalnya mengklik tombol), event `ActionEvent` dipicu, dan setiap `ActionListener` yang terdaftar pada komponen tersebut akan menerima notifikasi. Contoh Penerapan:

Ketika pengguna mengklik tombol "Clear All" di antarmuka SelesaiParkir, `ActionListener` yang terdaftar untuk tombol ini akan mendeteksi event `ActionEvent`. Langkah selanjutnya adalah menjalankan kode yang sesuai untuk menghapus data parkir dari model (misalnya, mengubah data pada objek parkir yang terkait). Setelah perubahan dilakukan pada model (data parkir), view (antarmuka pengguna) perlu diperbarui untuk mencerminkan perubahan tersebut. Dengan menggunakan pola Observer, komponen-komponen yang mengamati (observer) model akan diberitahu secara otomatis tentang perubahan ini, sehingga mereka dapat memperbarui tampilan dengan informasi yang tepat.

### 2.3. Design Principle

1. Single Responsibility Principle (SRP): Setiap kelas atau objek harus memiliki satu tanggung jawab utama saja untuk memisahkan tanggung jawab dan menjaga kode tetap sederhana dan mudah dipelihara. Contohnya ada di Kelas Akun hanya bertanggung jawab untuk menangani data pengguna (username dan password), sehingga mengikuti prinsip SRP dengan baik.
2. Open/Closed Principle (OCP): Kelas atau objek harus terbuka untuk perluasan tetapi tertutup untuk modifikasi. Perubahan dilakukan dengan menambahkan kode baru tanpa mengubah kode yang ada. Contohnya yaitu Kelas Parkir dapat diperluas dengan menambahkan metode baru atau properti baru tanpa mengubah kode yang ada, mendukung OCP.
3. Liskov Substitution Principle (LSP): Objek dari kelas induk harus bisa digantikan dengan objek dari subkelasnya tanpa mengubah fungsionalitas program. Contoh:

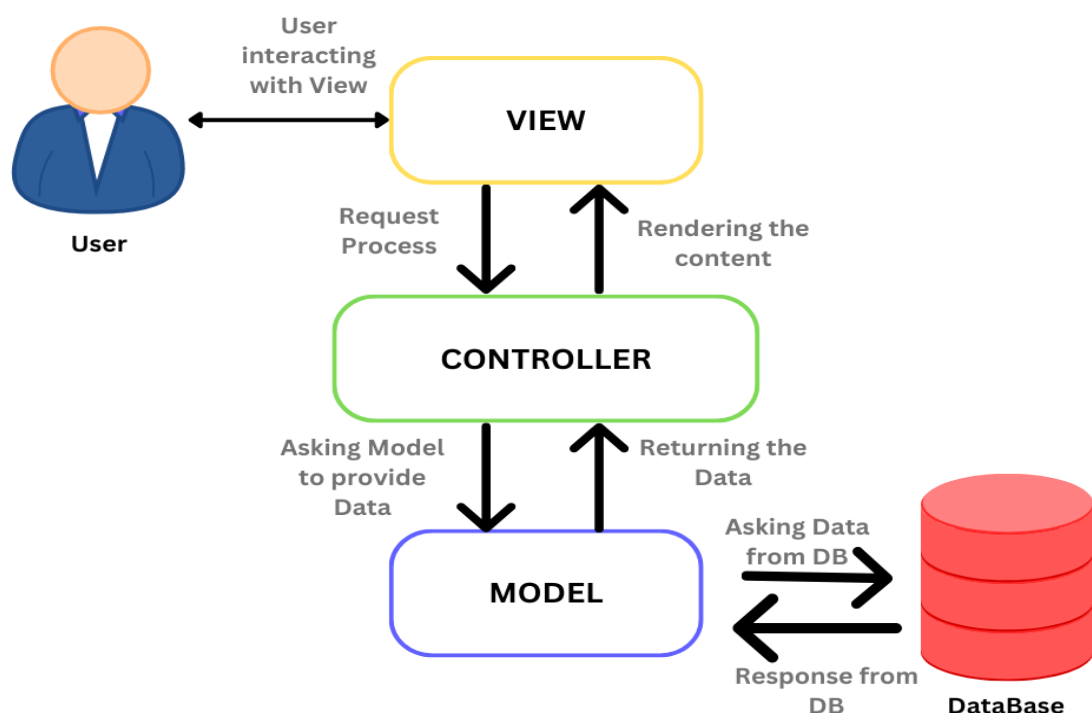


Subkelas dari Database harus bisa menggantikan Database tanpa mengubah fungsionalitas.

4. Interface Segregation Principle (ISP): Antarmuka yang besar dan kompleks harus dipecah menjadi antarmuka yang lebih kecil dan spesifik untuk menghindari ketergantungan yang tidak diperlukan. Contoh: Memecah antarmuka akses admin menjadi antarmuka untuk menghapus dan mengeluarkan parkir kendaraan.
5. Dependency Inversion Principle (DIP): Modul tingkat tinggi tidak harus bergantung pada modul tingkat rendah; keduanya harus bergantung pada abstraksi. Abstraksi tidak boleh bergantung pada detail; detail harus bergantung pada abstraksi. Contoh: Menggunakan antarmuka IDatabase agar MainFrame bergantung pada IDatabase daripada Database langsung.

#### 2.4. Model View Controller (MVC)

MVC Design Pattern (Model-View-Controller) adalah salah satu penerapan compound design pattern yang memisahkan kelas-kelas yang merepresentasikan data (model) dari kelas-kelas yang mengatur tampilan program (view) dan kelas-kelas pengendali data (controller). Meskipun kelas-kelas ini dibangun secara terpisah, mereka tetap terhubung satu sama lain.



Berdasarkan dengan skema diatas, terdapat empat package, yaitu view, controller, model, dan database.

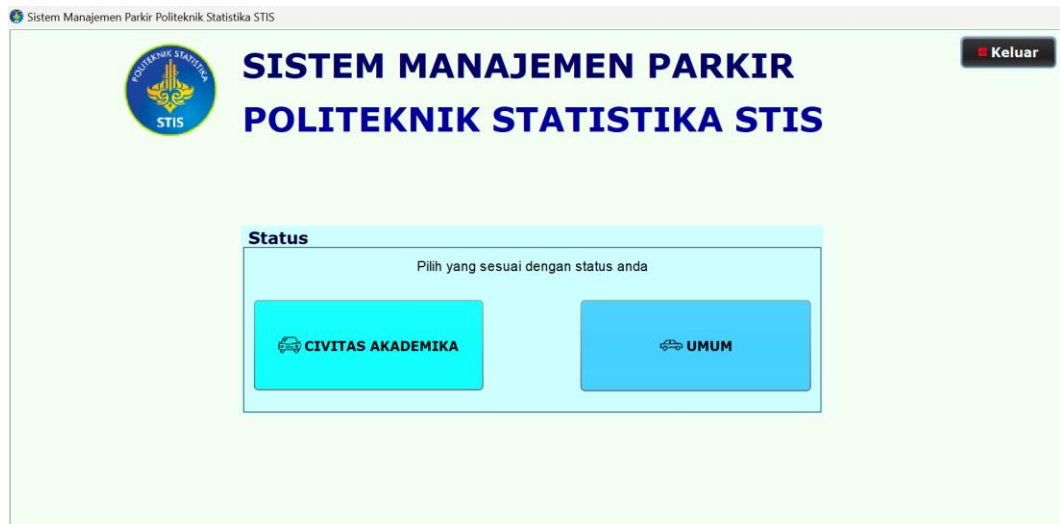
1. View: Kelas-kelas yang mengimplementasikan antarmuka pengguna dari program yang dibangun. Bagian ini yang dilihat oleh pengguna dan menjadi media interaksi pengguna. Setiap permintaan atau aksi yang diminta oleh pengguna yang berkaitan dengan data akan diteruskan ke bagian controller yang sesuai. Pada aplikasi ini, bagian view mencakup semua kelas kecuali kelas Akun, Parkir, Selesai, dan Database.
2. Controller: Kelas-kelas yang mengendalikan alur program secara keseluruhan, mengandung business logic, dan berfungsi sebagai penghubung antara view dengan model dan dao. Kelas-kelas ini merespon permintaan atau aksi dari view ke kelas dao yang diinginkan. Bagian controller yang digunakan pada aplikasi ini terletak pada setiap kelas yang memiliki inputan dari pengguna seperti kelas LoginAdmin, LoginCivitas, LoginUmum, dan RegisterPanel.
3. Model: Kelas-kelas yang menggambarkan struktur data dalam database. Bagian ini direpresentasikan oleh POJO (Plain Old Java Object), yaitu kelas Java biasa yang memiliki properti (atribut) dan method getter-setter untuk atribut-atributnya. Biasanya tidak memiliki method lain selain getter-setter. Pada aplikasi ini, bagian model mencakup kelas Akun, Parkir, dan Selesai.
4. Database: Kelas yang bertugas membuka koneksi dengan database tertentu dan memberikan objek koneksi ke kelas DAO (Data Access Object) yang membutuhkan koneksi. Pada aplikasi ini, bagian database juga diletakkan dalam kelas Database yang berisi method manipulatif yaitu Create, Read, Update, atau Delete CRUD data yang terhubung dengan Model.

## BAB 3

### ISI DAN DOKUMENTASI

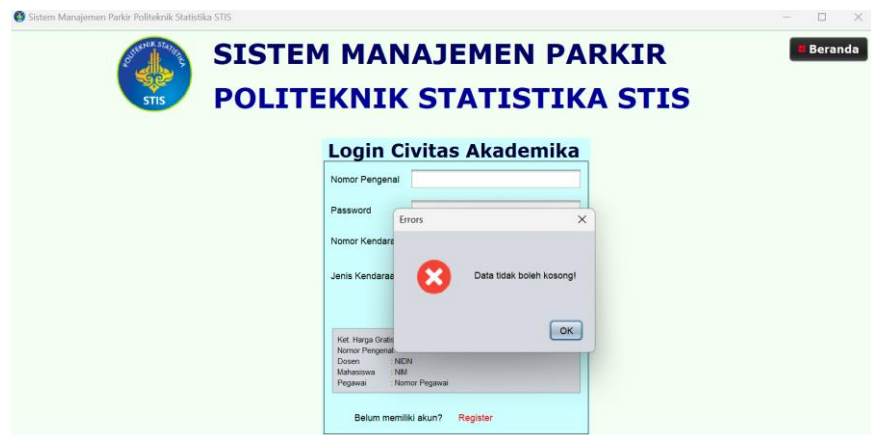
#### 3.1. Hasil Pengujian

1. Ketika user memilih Civitas Akademika

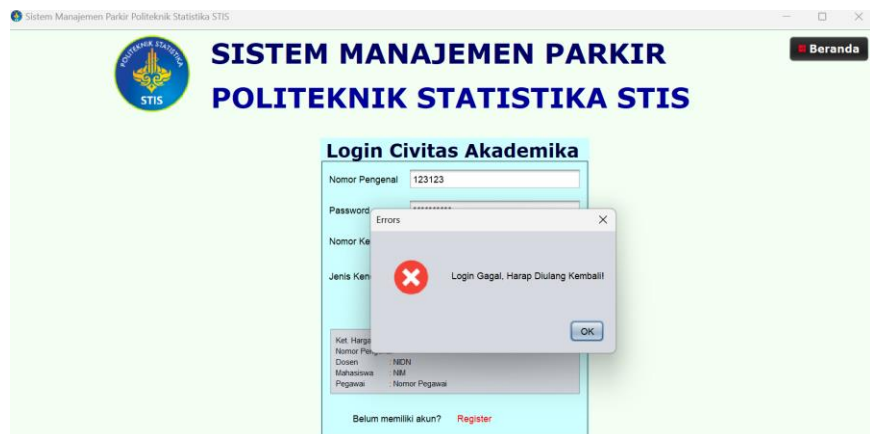


- a. Testing pada Form Login Civitas Akademika

- i. Jika data kosong



ii. Jika tidak berhasil login sebagai civitas akademika

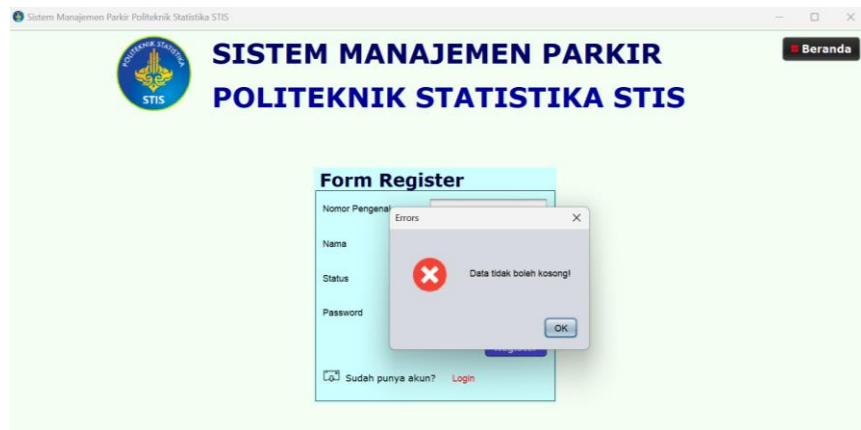


iii. Jika berhasil login sebagai civitas akademika

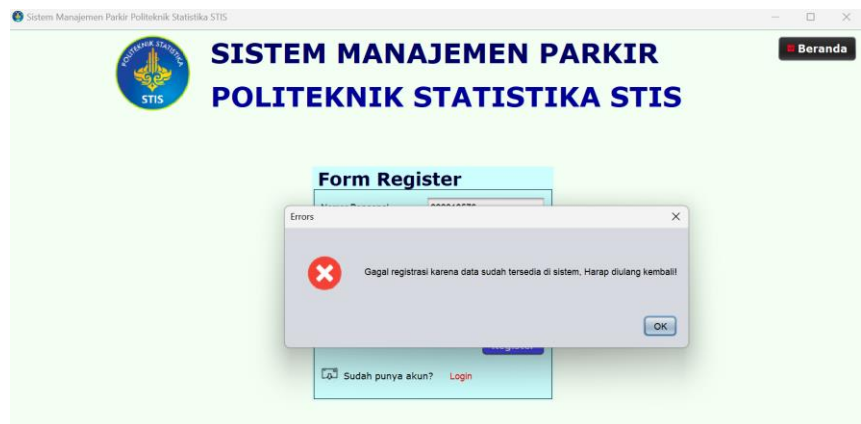


b. Testing pada Form Register

i. Jika data kosong



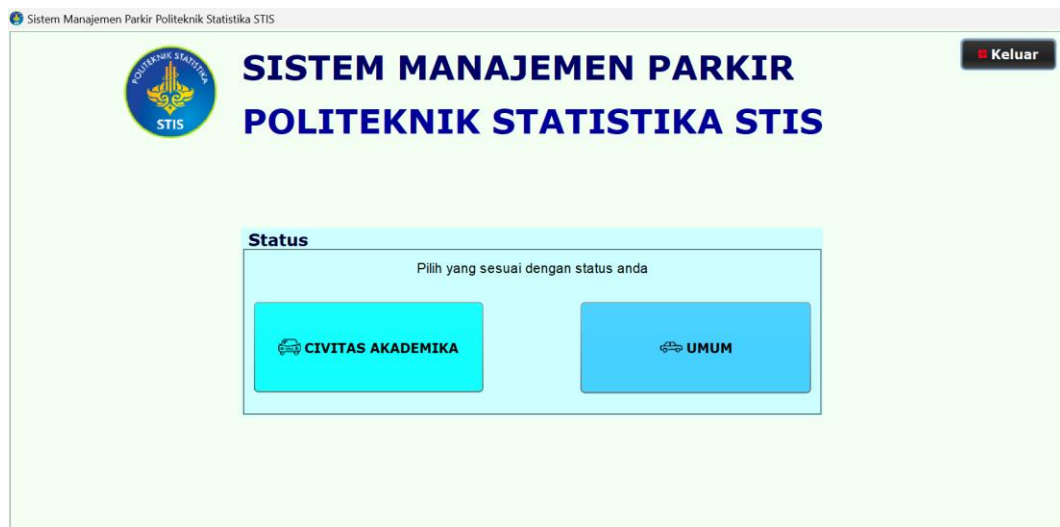
ii. Jika tidak berhasil register sebagai civitas akademika karena menggunakan data yang sama di database



iii. Jika berhasil register sebagai civitas akademika karena menggunakan data yang sama di database

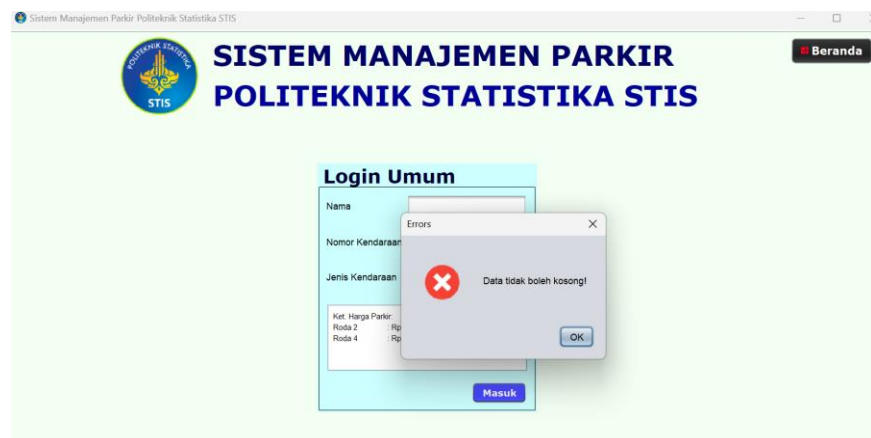


## 2. Ketika user memilih Umum

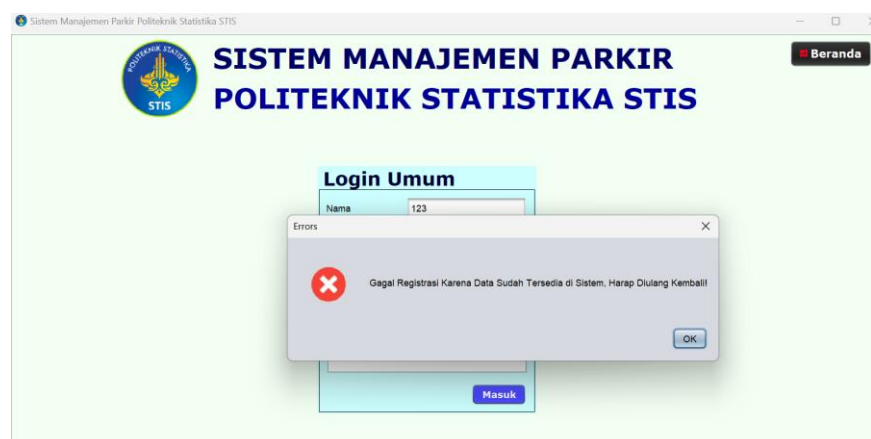


### a. Testing pada Form Login Umum

#### i. Jika data kosong



#### ii. Jika tidak berhasil login umum karena menggunakan data yang sama di database



iii. Jika berhasil login umum

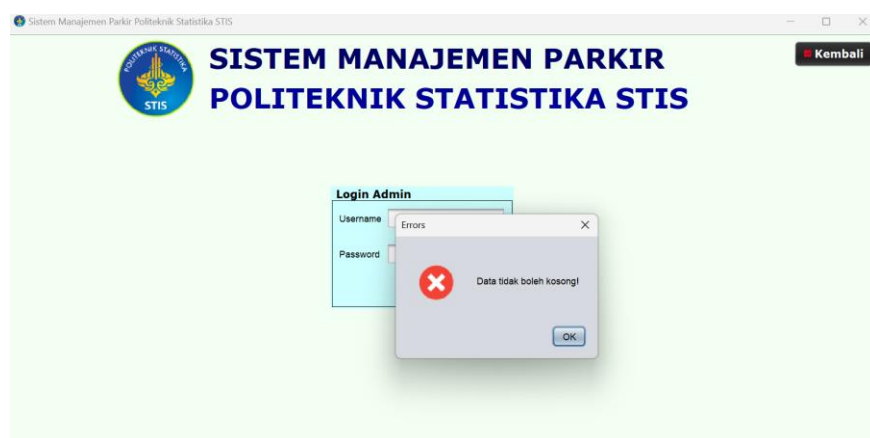


3. Ketika user memilih Login Admin

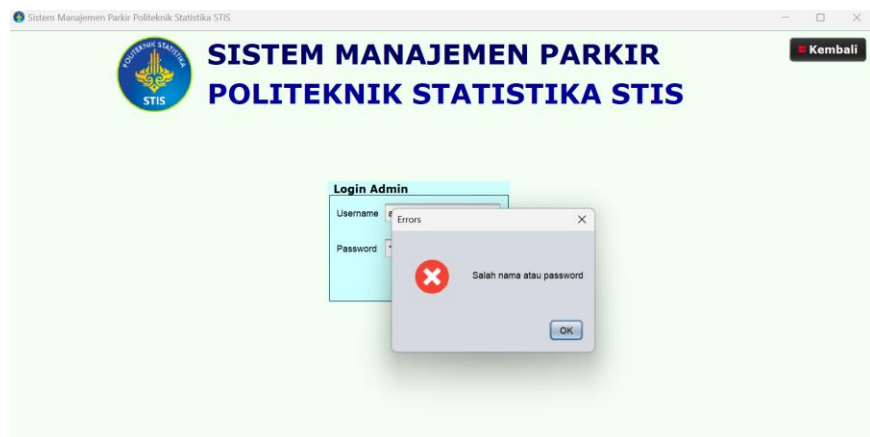


a. Testing pada Login Admin

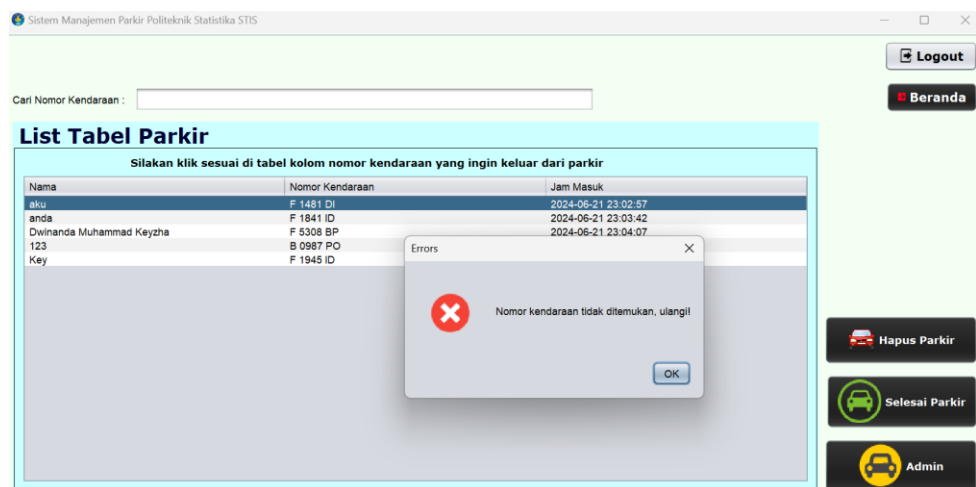
i. Jika data kosong



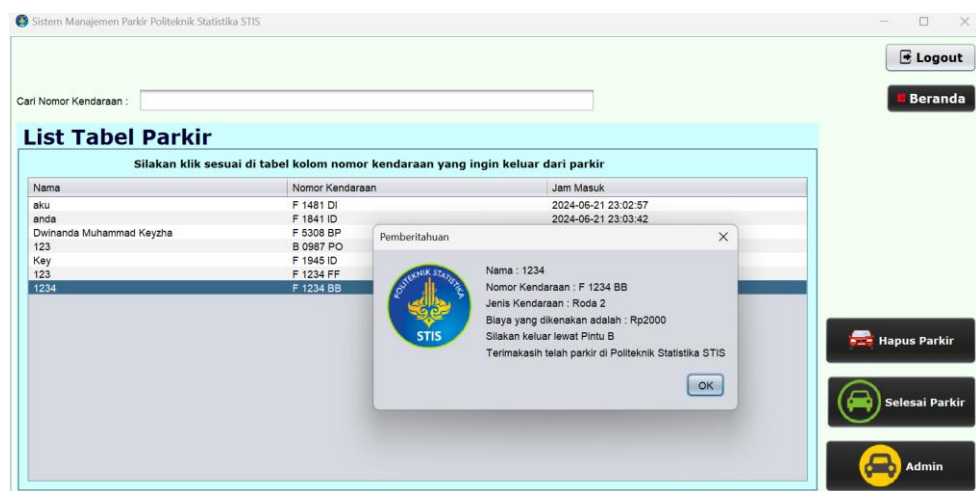
ii. Jika salah memasukkan username atau password



b. Menu Admin Keluar Parkir, jika admin salah klik

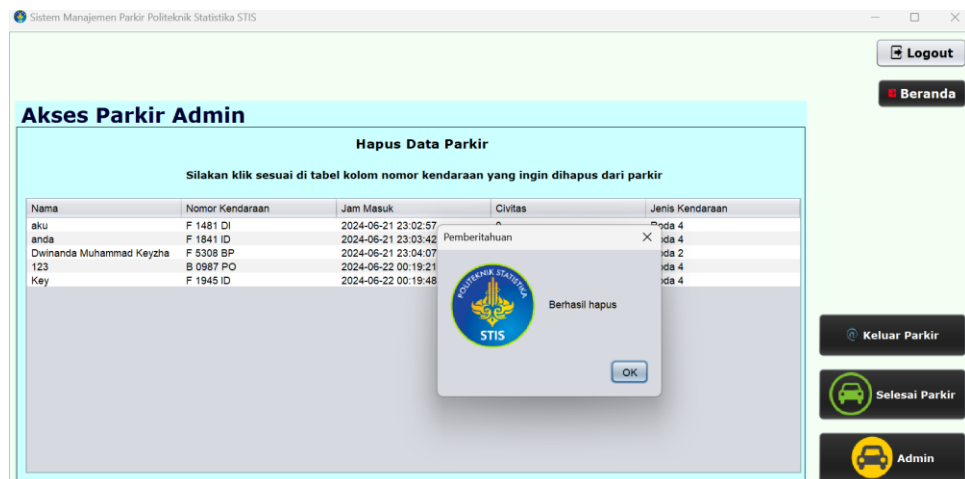


c. Menu Admin Keluar Parkir, jika admin berhasil klik



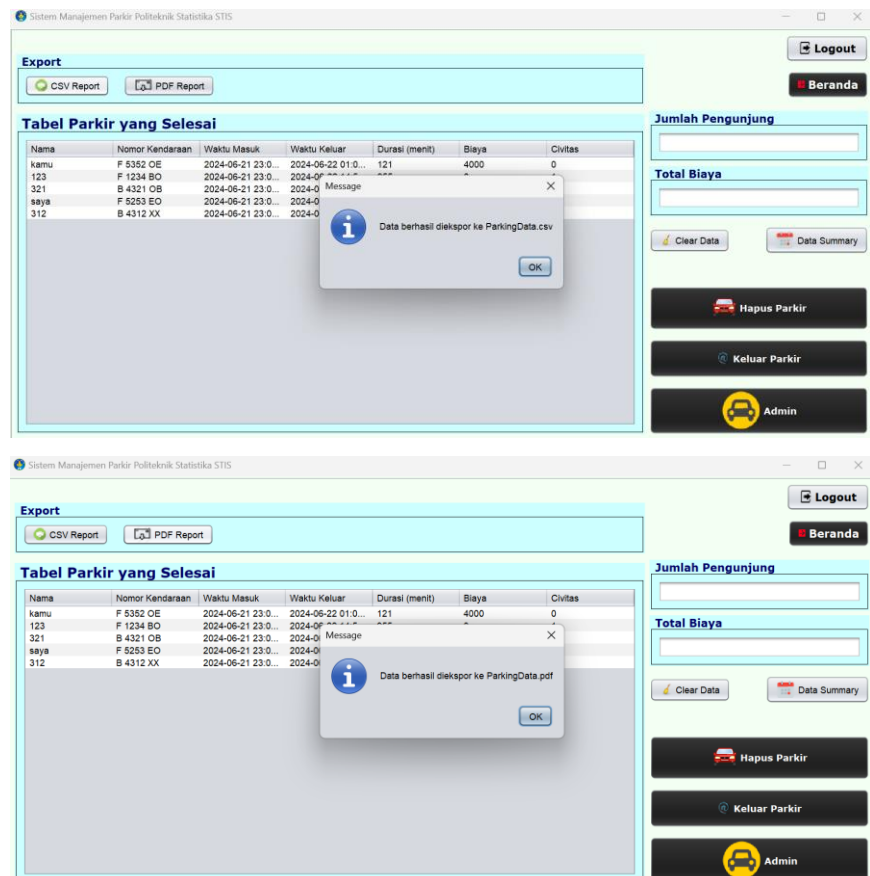


d. Menu Admin Hapus Parkir, jika admin berhasil klik

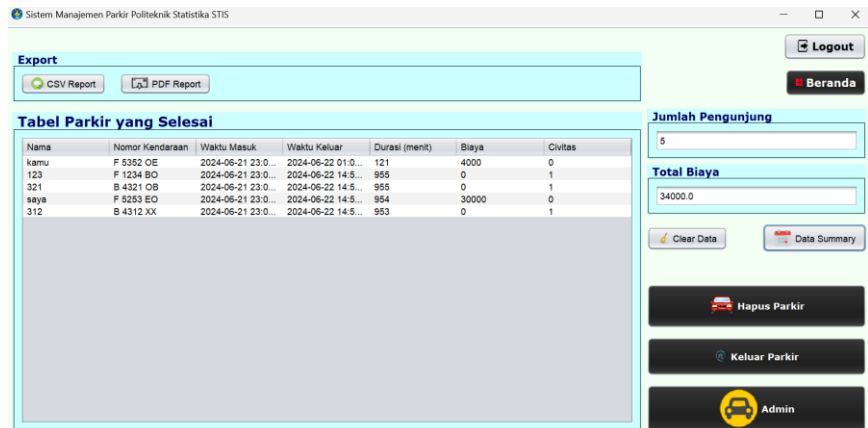


e. Testing pada Menu Admin Selesai Parkir

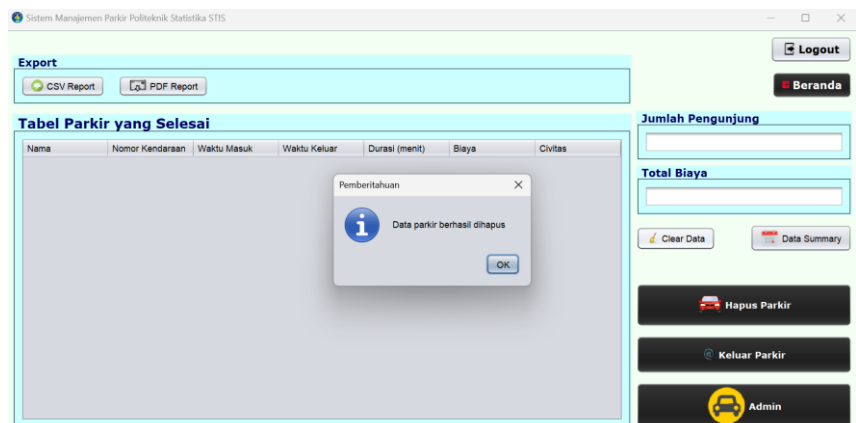
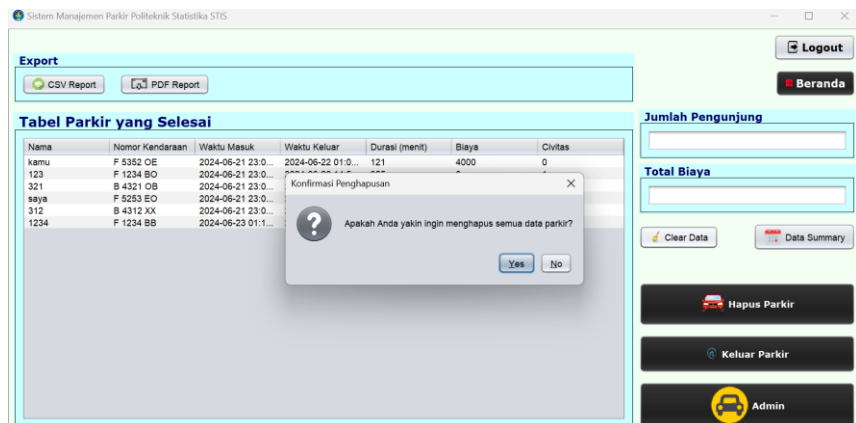
i. Jika berhasil export data ke csv atau pdf



ii. Jika berhasil mengklik data summary



iii. Jika berhasil mengklik clear data



### 3.2. Panduan Penggunaan dan Penjelasan Program

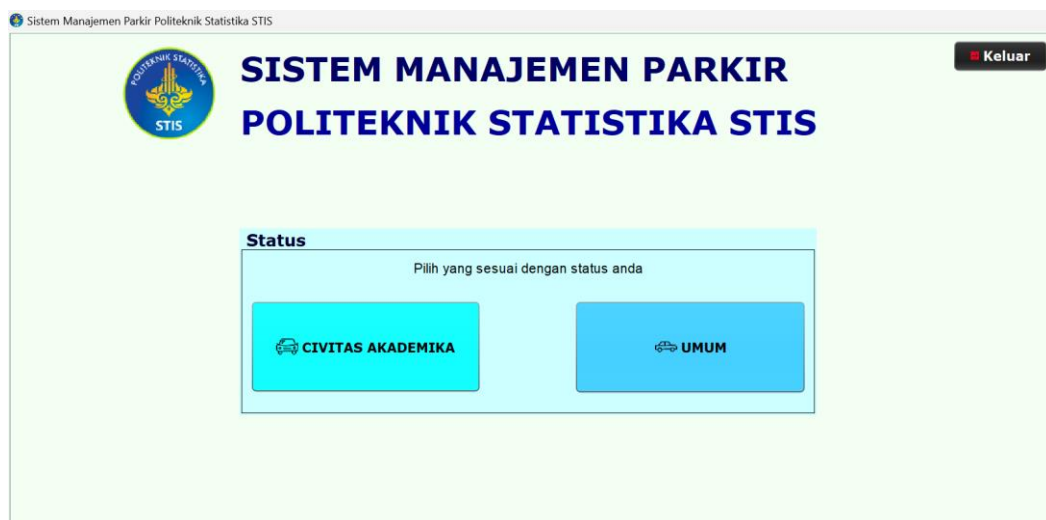
#### 1. Tampilan Awal (Home)



Gambar diatas adalah tampilan awal pada saat aplikasi dibuka. Jika pengguna ingin masuk, mereka bisa klik tombol masuk. Disini juga terdapat tombol login admin yang hanya bisa diakses oleh admin. Mereka bertindak sebagai controller, mengelola alur aplikasi berdasarkan interaksi pengguna. Home adalah bagian dari View, bertanggung jawab untuk menampilkan antarmuka pengguna.

Home memiliki design principle berupa Single Responsibility Principle (Prinsip Tanggung Jawab Tunggal) yaitu setiap kelas dan metode memiliki tanggung jawab yang jelas. Home bertanggung jawab atas tampilan awal aplikasi.

#### 2. Tampilan Beranda (BerandaParkir)



Gambar diatas adalah tampilan beranda untuk masuk, pengguna yang berstatus sebagai civitas akademika (dosen, mahasiswa, dan petugas/karyawan) dapat memilih tombol Civitas Akademika, sedangkan pengguna umum dapat memilih

tombol umum. BerandaParkir merupakan bagian dari View karena mengelola antarmuka pengguna. Penanganan aksi pada setiap button yang ada bertindak sebagai pengendali untuk memperbarui View berdasarkan interaksi pengguna. BerandaParkir hanya bertanggung jawab untuk mengelola tampilan beranda parkir sehingga memenuhi konsep Single Responsibility Principle.

### 3. Proses sebagai Civitas Akademika (Login Civitas Akademika)

Sistem Manajemen Parkir Politeknik Statistika STIS

**SISTEM MANAJEMEN PARKIR  
POLITEKNIK STATISTIKA STIS**

**Login Civitas Akademika**

Nomor Pengenal

Password

Nomor Kendaraan

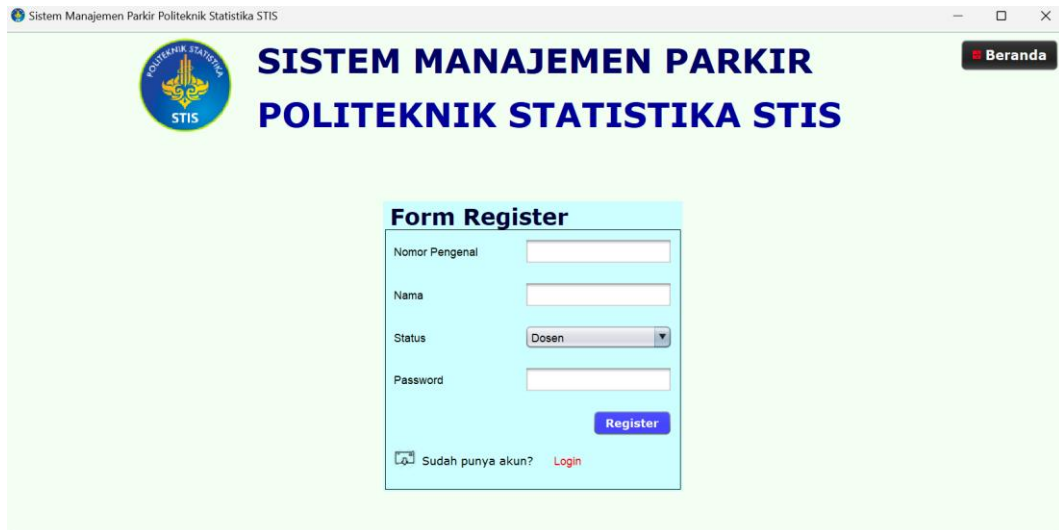
Jenis Kendaraan

Ket.	Harga Gratis
	Nomor Pengenal
Dosen	NICN
Mahasiswa	NIM
Pegawai	Nomor Pegawai

Belum memiliki akun? [Register](#)

Gambar diatas adalah tampilan untuk form login pengguna yang berstatus sebagai civitas akademika (dosen, mahasiswa, dan petugas/karyawan). Mereka yang sudah memiliki akun bisa langsung mengisi form dan melakukan login. Disini langsung di inputkan Nomor Kendaraan pengguna. Pada saat ini langsung ditangkap timestamp waktu pengguna parkir. Waktu ini akan digunakan untuk perhitungan biaya parkir. Namun untuk civitas akademika tidak akan dikenai biaya parkir alias gratis. Sedangkan jika belum memiliki akun, dapat melakukan register terlebih dahulu dengan mengklik tombol register. Dalam pola MVC, Model menangani akses data dan logika bisnis, seperti kelas Database yang berinteraksi dengan basis data/model parkir; View diwakili oleh kelas LoginCivitas yang mengelola antarmuka pengguna (GUI); sedangkan Controller adalah pendengar acara dalam kelas LoginCivitas yang memproses input dari pengguna. Prinsip SRP juga diterapkan, di mana setiap kelas memiliki tanggung jawab tunggal, misalnya LoginCivitas hanya menangani GUI login, sementara kelas lainnya mengurus aspek berbeda dari aplikasi.

#### 4. Tampilan Form Register (Register Civitas Akademik)



Gambar diatas adalah tampilan untuk form register yang bisa diisi oleh civitas akademika yang telah mempunyai nomor pengenal. Dapat terlihat isian apa saja yang harus di isi. Lalu mereka bisa kembali ke menu register tadi. Pola Model-View-Controller (MVC) yang diimplementasikan dalam aplikasi ini membagi tanggung jawab sesuai dengan prinsip-prinsip desain yang kuat. Model dalam MVC diwakili oleh kelas Database, yang bertanggung jawab atas pengelolaan data dan logika bisnis terkait interaksi dengan basis data terutama dengan kelas Akun. Ini mencakup operasi seperti validasi, penyimpanan, dan pengambilan data yang diperlukan oleh aplikasi. View, dalam hal ini direpresentasikan oleh RegisterPanel, bertanggung jawab menyediakan antarmuka pengguna yang memungkinkan civitas akademika untuk melakukan registrasi. Kontrol input dari pengguna dan pembaruan model dilakukan oleh kontroler, yang diimplementasikan melalui action listener dalam RegisterPanel. Ketika pengguna mengklik tombol "Register", kontroler ini memvalidasi input. RegisterPanel hanya fokus pada tugasnya sebagai antarmuka pengguna untuk registrasi. Single Responsibility Principle memastikan bahwa setiap kelas memiliki fokus yang jelas dan tidak melakukan terlalu banyak hal dalam satu waktu. Dengan menerapkan Open/Closed Principle (OCP), desain aplikasi ini terbuka untuk ekstensi dengan menambahkan fitur baru seperti validasi tambahan atau jenis pengguna baru tanpa mengubah inti dari kode yang sudah ada. Ini memungkinkan fleksibilitas dalam pengembangan dan pemeliharaan aplikasi secara keseluruhan.

## 5. Proses Sebagai Umum (Login Umum)

Sistem Manajemen Parkir Politeknik Statistika STIS

**SISTEM MANAJEMEN PARKIR  
POLITEKNIK STATISTIKA STIS**

**Login Umum**

Nama

Nomor Kendaraan

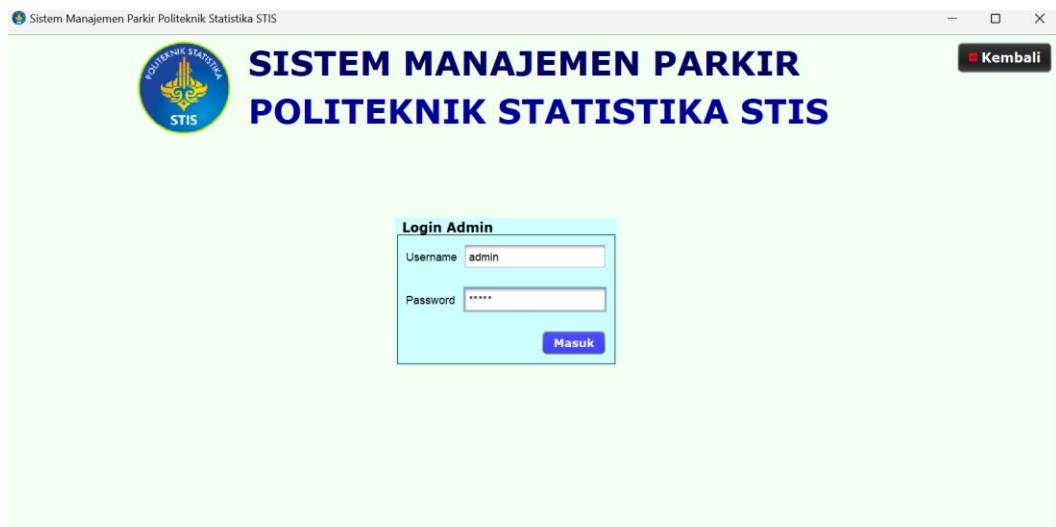
Jenis Kendaraan

Ket. Harga Parkir:  
Roda 2 : Rp2000/Jam  
Roda 4 : Rp4000/Jam

**Masuk**

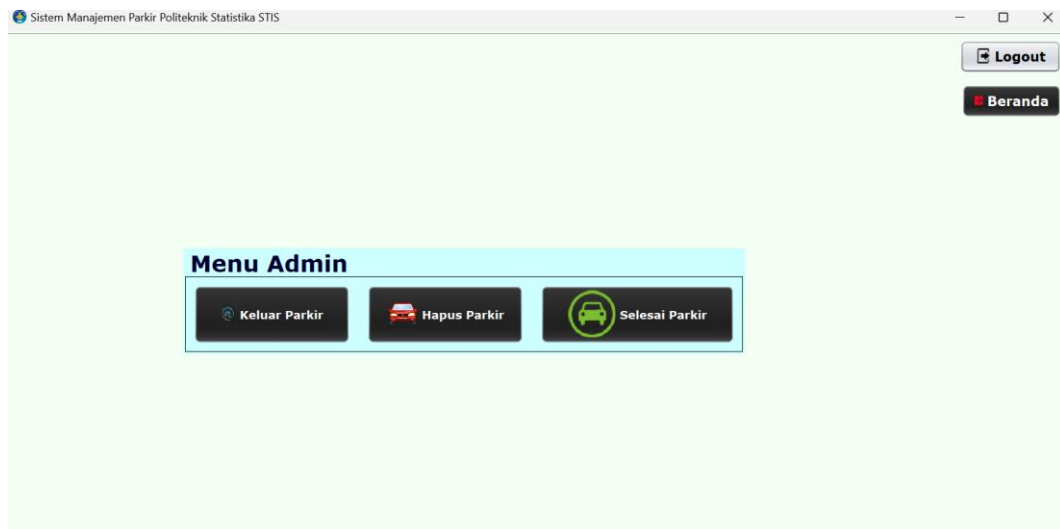
Gambar diatas adalah tampilan untuk login jika pengguna adalah umum. Mereka cukup mengisi nama, nomor kendaraan, dan jenis kendaran. Ketika login, ditangkap waktu login yang akan digunakan untuk perhitungan biaya. Biaya per jam untuk kendaraan roda 2 adalah Rp2000/jam dan kendaraan roda 4 adalah Rp4000/jam. Konsep Model-View-Controller (MVC) diterapkan dengan baik dalam aplikasi ini. Model, yang diwakili oleh kelas Database.java, bertanggung jawab atas manajemen interaksi dengan basis data, termasuk pengecekan duplikat data dengan method cekDuplikatUmum dan penyimpanan data masuk parkir dengan entriMasukParkirUmum. View, yang terdiri dari kelas LoginUmum.java, menyediakan antarmuka pengguna yang intuitif dengan komponen-komponen UI. Controller, juga di dalam kelas LoginUmum.java, bertanggung jawab untuk mengatur input pengguna, melakukan validasi data, dan memperbarui model sesuai dengan interaksi pengguna sebelum data disimpan ke basis data. Single Responsibility Principle diterapkan dengan baik di mana setiap kelas hanya memiliki tanggung jawab khusus, baik untuk menampilkan UI atau untuk mengelola data dan logika bisnis. Open/Closed Principle (OCP) juga terpenuhi dengan baik, memungkinkan aplikasi untuk diperluas dengan penambahan fitur baru tanpa harus memodifikasi kode yang sudah ada.

## 6. Proses Admin (Login Admin)



Gambar diatas adalah tampilan untuk login admin. Admin bisa mengakses beberapa fitur penting untuk sistem manajemen parkir. View pada tampilan ini terdiri dari komponen-komponen UI. Tampilan dibuat dengan menggunakan Java Swing untuk menyediakan antarmuka pengguna yang memungkinkan admin untuk memasukkan username dan password untuk login. UI ini dirancang secara sederhana dan intuitif. Controller juga terletak di dalam kelas LoginAdmin.java. Di sini, kita memiliki loginAdminButtonActionPerformed yang berfungsi sebagai action listener untuk tombol "Masuk". Di dalam method ini, validasi dilakukan terhadap input yang dimasukkan admin (nama dan password). Jika input kosong, akan muncul pesan error. Jika input admin ("admin" sebagai username dan "admin" sebagai password) sesuai, tampilan akan dialihkan ke MenuAdmin melalui contentScrollPane. Jika tidak sesuai, pesan kesalahan akan ditampilkan. Model mencakup kelas Akun yang mengelola data akun admin, seperti nama, nomor pengenal, dan status. Ini mencerminkan entitas data yang terkait dengan pengguna admin dalam sistem. Single Responsibility Principle juga diterapkan dengan Setiap kelas, baik itu View (LoginAdmin.java), Controller (LoginAdmin.java), atau Model (Akun.java), hanya memiliki satu tanggung jawab khusus.

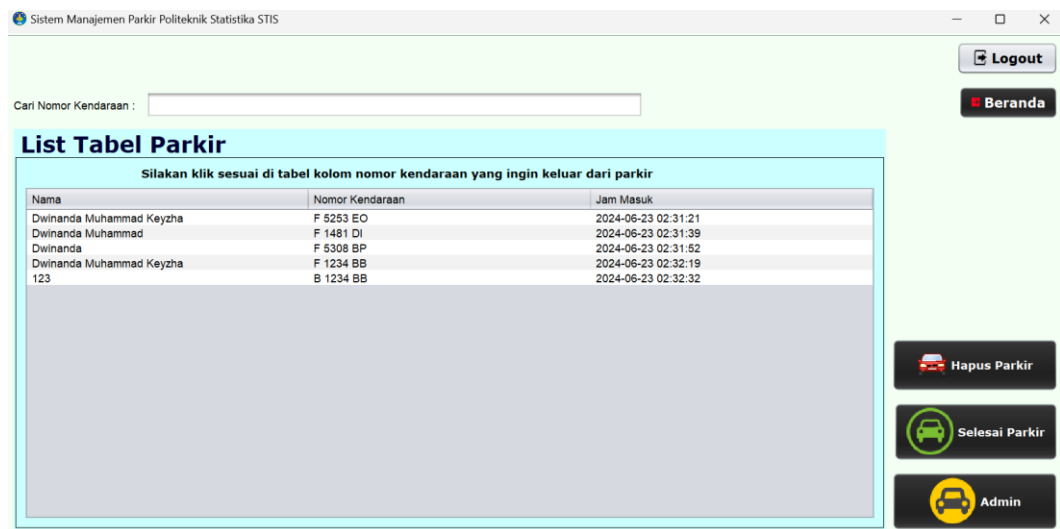
## 7. Tampilan Menu Admin (Menu Admin)



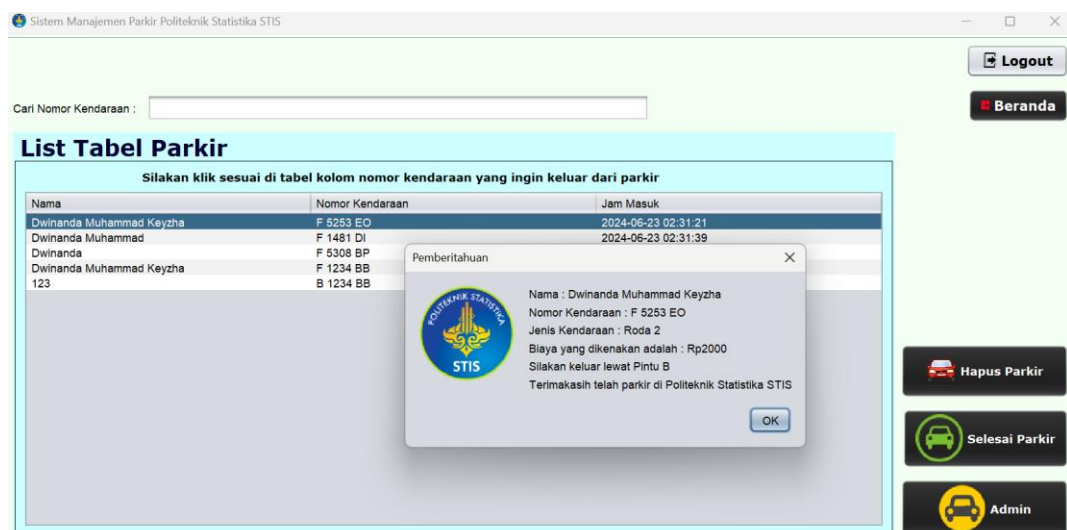
Gambar diatas adalah tampilan untuk menu admin setelah berhasil login sebagai admin. Menu Admin ini menyediakan akses ke beberapa fitur penting dalam sistem manajemen parkir. Admin dapat melakukan berbagai operasi seperti menghapus data parkir, menandai keluar parkir kendaraan, dan melakukan beberapa hal untuk data parkir yang sudah selesai. Dalam implementasi aplikasi ini, konsep Model-View-Controller (MVC) dijalankan dengan baik. View-nya, yang terwakili oleh kelas MenuAdmin (MenuAdmin.java), bertanggung jawab atas tampilan antarmuka pengguna (UI). Ini mencakup berbagai tombol dan panel yang memungkinkan admin untuk melakukan operasi tertentu dalam sistem manajemen parkir. Kontroller, yang juga diimplementasikan dalam MenuAdmin.java, mengatur aksi-aksi pengguna dan logika bisnis terkait. Setiap tombol pada antarmuka memiliki ActionListener yang khusus mengelola interaksi pengguna, seperti memperbarui tampilan panel di contentScrollPane untuk menampilkan halaman yang sesuai dengan aksi yang dilakukan. Prinsip-prinsip desain yang diterapkan dalam aplikasi ini juga mencakup Single Responsibility Principal yang jelas. Desain mematuhi Open/Closed Principle dengan baik. Hal ini memungkinkan untuk menambahkan fitur baru ke dalam aplikasi atau mengubah perilaku tombol-tombol aksi tanpa perlu melakukan perubahan besar-besaran pada kode yang sudah ada. Dengan demikian, aplikasi ini tidak hanya berfungsi sesuai dengan kebutuhan, tetapi juga dirancang untuk dapat dikelola dan dikembangkan dengan efisien ke depannya.



## 8. Proses Admin Keluar Parkir (Keluar Parkir)



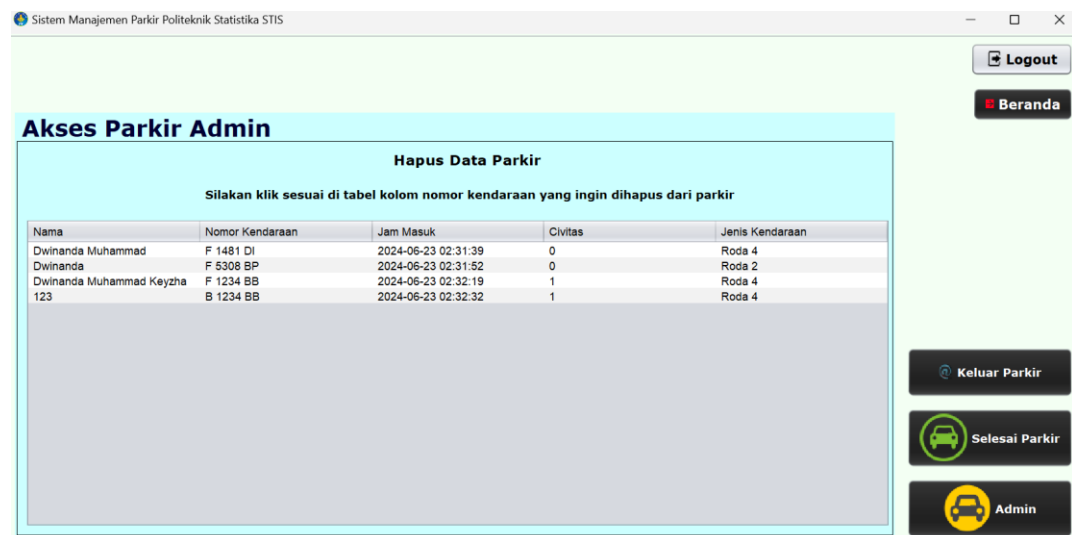
Gambar diatas adalah tampilan untuk semua kendaraan yang sedang parkir. Tabel yang tersedia menampilkan informasi parkir seperti nama, nomor kendaraan, dan waktu masuk. Pengguna dapat mencari nomor kendaraan menggunakan fitur pencarian. Ketika pengguna mengklik nomor kendaraan pada tabel, pengguna dapat menghapus data parkir tersebut atau menyelesaikan proses parkir. Selanjutnya akan otomatis generate tarif parkir sesuai dengan durasi parkir.



Dalam aplikasi ini, konsep Model-View-Controller (MVC) diterapkan secara sistematis. Model direpresentasikan oleh kelas Parkir, yang mencerminkan entitas parkir dengan atribut seperti nama, nomor kendaraan, waktu masuk, dan jenis kendaraan. Data parkir diambil dari database menggunakan metode `Database.getInstance().dataTable()`. View diimplementasikan melalui `KeluarParkir.java`, yang bertanggung jawab sebagai antarmuka pengguna (UI) untuk proses keluar parkir. Komponen UI seperti tabel, tombol, dan panel dibuat

menggunakan Java Swing. Fitur pencarian (filter) di dalam tabel juga ditangani di dalam view ini, memungkinkan pengguna untuk mencari nomor kendaraan dengan mudah. Controller mengontrol aksi pengguna dengan mengimplementasikan ActionListener untuk tombol-tombol dan interaksi lainnya. Metode keluarTableMouseClicked mengelola proses ketika pengguna mengklik pada tabel untuk proses keluar parkir, sedangkan filterTable() digunakan untuk menyaring data di tabel berdasarkan input pengguna. Prinsip Single Responsibility Principle diterapkan dengan setiap metode memiliki tanggung jawab spesifik, seperti mengatur tampilan, mengelola data, atau menanggapi interaksi pengguna. Desain juga mematuhi Open/Closed Principle, memungkinkan penambahan fitur baru tanpa harus memodifikasi kode yang sudah ada secara drastis, yang tercermin dalam fleksibilitas penggunaan ActionListener dan pengelolaan tampilan baru di contentScrollPane.

## 9. Proses Admin Hapus Parkir



Gambar diatas adalah tampilan untuk semua kendaraan yang sedang parkir. Pengguna yang ingin dihapus karena mungkin merasa bermasalah silakan untuk mengklik Nomor Kendaraan yang sesuai dengan kendaraannya. Model dalam aplikasi ini diwakili oleh kelas Parkir, yang berfungsi sebagai entitas utama yang mencerminkan atribut-atribut penting seperti nama, nomor kendaraan, waktu masuk, jenis entitas, dan jenis kendaraan. Akses data dilakukan melalui metode Database.getInstance().dataTable(), yang mengambil data parkir dari database secara terenkapsulasi. View diimplementasikan dalam kelas HapusParkir, yang bertindak sebagai antarmuka pengguna (UI) menggunakan Java Swing. View mengelola interaksi pengguna seperti mengklik baris dalam tabel untuk menghapus

data parkir (`hapusTableMouseClicked`), serta mengatur aksi tombol seperti menyelesaikan parkir, kembali ke beranda admin, atau keluar dari sesi admin. Controller terdiri dari implementasi `ActionListener` yang menangani respons terhadap tindakan pengguna pada tombol dan interaksi UI lainnya. Controller merespons aksi pengguna dengan memuat tampilan baru atau melakukan operasi lainnya seperti penghapusan data parkir melalui `hapusTableMouseClicked`. Setiap kelas memiliki tanggung jawab spesifik sesuai dengan `Single Responsibility Principle`, memastikan bahwa setiap metode dan kelas fokus pada satu aspek tugasnya. Desain juga mendukung `Open/Closed Principle` dengan memungkinkan penambahan fitur baru seperti tombol dan elemen UI tanpa perlu memodifikasi kode yang ada secara signifikan, melainkan dengan mengintegrasikan fitur baru ke dalam struktur yang sudah ada.

#### 10. Proses Admin Selesai Parkir

**Export**

CSV Report PDF Report

**Tabel Parkir yang Selesai**

Nama	Nomor Kendaraan	Waktu Masuk	Waktu Keluar	Durasi (menit)	Biaya	Civitas
anda	F 1841 ID	2024-06-21 23:0...	2024-06-23 01:1...	1572	104000	0
Dwinanda Muha...	F 5308 BP	2024-06-21 23:0...	2024-06-23 01:1...	1572	0	1
123	B 0987 PO	2024-06-22 00:1...	2024-06-23 01:1...	1496	0	1
Key	F 1945 ID	2024-06-22 00:1...	2024-06-23 01:1...	1496	96000	0
123	F 1234 FF	2024-06-23 01:0...	2024-06-23 01:1...	60	0	1
Dwinanda Muha...	F 5253 EO	2024-06-23 02:3...	2024-06-23 02:3...	60	2000	0

**Jumlah Pengunjung**

**Total Biaya**

Clear Data Data Summary

Hapus Parkir

Keluar Parkir

Admin

Gambar diatas adalah tampilan untuk semua kendaraan yang sudah selesai parkir. Terdapat berbagai fungsionalitas terkait seperti ekspor ke CSV dan PDF, serta statistik jumlah pengunjung dan total biaya parkir. Model mewakili entitas data seperti Selesai dan Database, di mana Selesai merepresentasikan data parkir yang sudah selesai dan Database bertanggung jawab atas akses ke data dari database. View diimplementasikan dalam kelas `SelesaiParkir`, yang bertindak sebagai antarmuka pengguna (UI). Kelas `SelesaiParkir` juga berfungsi sebagai controller dalam pola MVC ini. Ini mengatur aksi-aksi pengguna seperti ekspor data, perhitungan statistik, dan navigasi antar halaman menggunakan tombol seperti Beranda, Admin, dan Keluar Parkir. Dengan demikian, pola MVC memisahkan tegas antara logika bisnis aplikasi (Model), tampilan pengguna (View), dan

pengontrol interaksi pengguna (Controller), memfasilitasi pengembangan dan pemeliharaan aplikasi yang lebih terstruktur dan terpisah. Dalam hal prinsip desain, aplikasi ini memperlihatkan penerapan prinsip Single Responsibility Principle (SRP) dengan jelas. Kelas SelesaiParkir bertanggung jawab hanya terhadap tampilan dan pengendalian interaksi pengguna, memungkinkan untuk memodifikasi antarmuka pengguna tanpa mempengaruhi logika bisnisnya. Hal ini meningkatkan modularitas dan kemudahan perawatan. Open/Closed Principle (OCP) juga terdapat implementasinya, konsep ini mendorong pemisahan yang baik antara bagian-bagian aplikasi yang cenderung berubah dan bagian-bagian yang stabil, mendukung perluasan aplikasi dengan lebih baik di masa depan.

## **LINK VIDEO**

[https://s.stis.ac.id/222212576 DwinandaMuhammadKeyzha UASPBO](https://s.stis.ac.id/222212576_DwinandaMuhammadKeyzha_UASPBO)

## **LINK GIT REPOSITORY**

<https://github.com/dwinandakey/Projek-Akhir-UAS-PBO.git>