# 1TiketKeretaApi - Sistem Pemesanan Tiket Kereta Api

## JAR yang Dapat Dieksekusi

dist/1TiketKeretaApi.jar

## Layar Awal (Login)

* Form login dengan kolom:
  + Username
  + Password
* Autentikasi dilakukan terhadap tabel admin di database.

## Navigasi Utama (tiket.Home)

Setelah login berhasil, pengguna diarahkan ke jendela utama dengan fitur berikut:

### Formulir Pemesanan Tiket

* Kolom input:
  + ID Penumpang
  + Nama Penumpang
  + Alamat
  + Stasiun Awal (default: Bandung - BD)
  + Stasiun Tujuan
  + Kereta (default: Argo Wilis, Argo Parahyangan, Lodaya)
  + Gerbong (default: Eksekutif 1/2, Ekonomi 1/2/3)
  + Tanggal Keberangkatan
  + Jumlah Penumpang Dewasa dan Anak-anak

### Perhitungan Harga

* Harga tiket dihitung secara dinamis berdasarkan:
  + Status (Dewasa/Anak)
  + Kelas Gerbong
* Data harga diambil dari tabel harga di database.

### Pembuatan Tiket

* Data pemesanan disimpan ke tabel penumpang.
* Tiket dicetak menggunakan JasperReports (Tiket.jasper).

## Bilah Menu

### Menu File

* **Admin**: Akses ke pengelolaan admin dan harga.
* **Keluar**: Menutup aplikasi.

### Menu Data Penumpang

* Akses ke tampilan dan pengelolaan data penumpang.

## Panel Admin (Admin.java)

Akses melalui: Menu File → Admin

### 1. Manajemen Pengguna Admin

* Fitur:
  + Tambah Admin
  + Edit Admin
  + Hapus Admin
* Kolom:
  + Nama
  + Username
  + Password
* Tampilan menggunakan JTable.
* Menggunakan JTextField bayangan1 untuk menyimpan ID tersembunyi saat edit/hapus.

### 2. Manajemen Harga Tiket

* Lihat dan edit harga tiket.
* Kolom:
  + Status (Dewasa/Anak)
  + Kelas Gerbong
  + Harga
* Menggunakan JTable dan tombol Simpan untuk menyimpan perubahan.

## Panel Data Penumpang (Penumpang.java)

Akses melalui: Menu Data Penumpang

### Fitur:

* Melihat semua data dari tabel penumpang.
* Kolom tampilan:
  + ID Penumpang
  + Nama
  + Alamat
  + Kereta
  + Gerbong
  + Tujuan
  + Waktu Berangkat / Tiba
  + Tanggal
  + Jumlah Dewasa / Anak
  + Total Harga
* Cetak Laporan Penumpang: Menggunakan penumpang.jasper.
* Hapus Semua Data Penumpang: Mengosongkan tabel penumpang setelah konfirmasi.

## Catatan Teknis

* Bahasa: Java (GUI menggunakan Swing)
* Database: MySQL
  + Koneksi ditangani oleh tiket.Koneksi
* Logika utama aplikasi: tiket.Home
* Laporan menggunakan JasperReports
* Driver Database: mysql-connector-java
* Menjalankan aplikasi di lingkungan tanpa GUI (headless) akan menyebabkan java.awt.HeadlessException

## Ringkasan Fitur Utama

| Fitur | Deskripsi |
| --- | --- |
| Login | Autentikasi ke sistem sebagai admin. |
| Pemesanan Tiket | Formulir lengkap dengan kalkulasi harga otomatis. |
| Cetak Tiket | Menyimpan dan mencetak tiket dalam format JasperReport. |
| Admin Panel | Kelola akun admin dan harga tiket. |
| Data Penumpang | Lihat, cetak laporan, dan hapus data pemesanan penumpang. |
| Navigasi GUI | Antar-frame: Home, Admin, dan Penumpang melalui menu. |

# Skenario Pengujian untuk Aplikasi 1TiketKeretaApi

## Skenario Login

**ID:** TC\_LOGIN\_001 **Judul:** Login berhasil dengan kredensial admin yang valid. **Prakondisi:** - Username dan password admin yang valid ada di tabel admin. - Aplikasi berada di layar Login. **Langkah-langkah:** 1. Masukkan username admin yang valid di kolom “Username”. 2. Masukkan password valid yang sesuai di kolom “Password”. 3. Klik tombol “LOGIN”. **Hasil yang Diharapkan:** - Jendela Login tertutup. - Jendela Beranda (pemesanan tiket) terbuka. **Hasil Aktual:** Kode di Login.java (btnloginActionPerformed) menjalankan kueri SQL. Jika kecocokan ditemukan, ia memanggil new Home().setVisible(true); dan dispose();. **Status:** Lulus

**ID:** TC\_LOGIN\_002 **Judul:** Login gagal dengan username tidak valid. **Prakondisi:** - Aplikasi berada di layar Login. **Langkah-langkah:** 1. Masukkan username yang tidak valid di kolom “Username”. 2. Masukkan password apa pun di kolom “Password”. 3. Klik tombol “LOGIN”. **Hasil yang Diharapkan:** - Pesan kesalahan “Gagal Login Oi” (atau serupa) ditampilkan. - Pengguna tetap berada di layar Login. **Hasil Aktual:** Kueri SQL SELECT \* FROM admin WHERE username='<username\_tidak\_valid>' And password='<password>' dijalankan. rs.next() bernilai false. JOptionPane.showMessageDialog(null, "Gagal Login Oi"); ditampilkan. Layar Login tetap ada. **Status:** Lulus

**ID:** TC\_LOGIN\_003 **Judul:** Login gagal dengan password tidak valid. **Prakondisi:** - Username admin yang valid ada di tabel admin. - Aplikasi berada di layar Login. **Langkah-langkah:** 1. Masukkan username admin yang valid di kolom “Username”. 2. Masukkan password yang tidak valid di kolom “Password”. 3. Klik tombol “LOGIN”. **Hasil yang Diharapkan:** - Pesan kesalahan “Gagal Login Oi” (atau serupa) ditampilkan. - Pengguna tetap berada di layar Login. **Hasil Aktual:** Kueri SQL SELECT \* FROM admin WHERE username='<username\_valid>' And password='<password\_tidak\_valid>' dijalankan. rs.next() bernilai false. JOptionPane.showMessageDialog(null, "Gagal Login Oi"); ditampilkan. Layar Login tetap ada. **Status:** Lulus

**ID:** TC\_LOGIN\_004 **Judul:** Login gagal dengan username kosong. **Prakondisi:** - Aplikasi berada di layar Login. **Langkah-langkah:** 1. Biarkan kolom “Username” kosong. 2. Masukkan password apa pun di kolom “Password”. 3. Klik tombol “LOGIN”. **Hasil yang Diharapkan:** - Pesan kesalahan yang menunjukkan kegagalan (misalnya, “Gagal Login Oi” atau kesalahan database spesifik jika kueri gagal) ditampilkan. - Pengguna tetap berada di layar Login. **Hasil Aktual:** Kueri SQL SELECT \* FROM admin WHERE username='' And password='<password>' dijalankan. Dengan asumsi tidak ada pengguna admin dengan username kosong, rs.next() bernilai false. JOptionPane.showMessageDialog(null, "Gagal Login Oi"); ditampilkan. Layar Login tetap ada. **Status:** Lulus

**ID:** TC\_LOGIN\_005 **Judul:** Login gagal dengan password kosong. **Prakondisi:** - Aplikasi berada di layar Login. **Langkah-langkah:** 1. Masukkan username apa pun di kolom “Username”. 2. Biarkan kolom “Password” kosong. 3. Klik tombol “LOGIN”. **Hasil yang Diharapkan:** - Pesan kesalahan yang menunjukkan kegagalan (misalnya, “Gagal Login Oi” atau kesalahan database spesifik jika kueri gagal) ditampilkan. - Pengguna tetap berada di layar Login. **Hasil Aktual:** Kueri SQL SELECT \* FROM admin WHERE username='<username>' And password='' dijalankan. Dengan asumsi pengguna yang ditentukan tidak memiliki password kosong, rs.next() bernilai false. JOptionPane.showMessageDialog(null, "Gagal Login Oi"); ditampilkan. Layar Login tetap ada. **Status:** Lulus

## Skenario Pemesanan Tiket (Layar Beranda)

**ID:** TC\_BOOK\_001 **Judul:** Pemesanan tiket berhasil dengan semua input valid untuk satu orang dewasa. **Prakondisi:** - Pengguna sudah login dan berada di layar Beranda. - Data harga untuk kereta/kelas yang dipilih ada di tabel harga. **Langkah-langkah:** 1. ID Penumpang (fid) terisi otomatis. 2. Masukkan nama penumpang di “Nama Penumpang”. 3. Masukkan alamat penumpang di “Alamat”. 4. “Stasiun Awal” adalah “Bandung (BD)”. 5. Pilih “Stasiun Tujuan” dari dropdown. 6. Pilih “Kereta Api” dari dropdown. 7. Pilih “Gerbong” dari dropdown. 8. Pilih tanggal dari “tanggal”. 9. Atur spinner “Dewasa” ke 1. 10. Atur spinner “Anak-anak” ke 0. 11. Verifikasi harga “Total” dihitung dan ditampilkan. 12. Masukkan jumlah yang valid di “Bayar” (lebih besar atau sama dengan Total). 13. Tekan Enter di kolom “Bayar” untuk menghitung “Kembali”. 14. Klik tombol “Simpan”. **Hasil yang Diharapkan:** - Pemesanan disimpan ke tabel penumpang. - Pesan sukses (atau proses pembuatan tiket) dimulai. - Jendela Beranda tertutup dan jendela Beranda baru (atau yang sama dikosongkan) ditampilkan dengan ID baru yang dibuat otomatis. - Proses pembuatan tiket JasperReports dipanggil (misalnya, JasperViewer.viewReport dipanggil). **Hasil Aktual:** Kode di Home.java (metode simpan) menjalankan kueri SQL INSERT. Jika berhasil, ia menutup jendela Home saat ini, membuat yang baru (melalui Index.oke()), lalu memanggil JasperFillManager.fillReport dan JasperViewer.viewReport dengan “src/report/Tiket.jasper” dan ID penumpang. **Status:** Lulus

**ID:** TC\_BOOK\_002 **Judul:** Pemesanan tiket berhasil dengan semua input valid untuk beberapa dewasa dan anak-anak. **Prakondisi:** - Pengguna sudah login dan berada di layar Beranda. - Data harga untuk kereta/kelas yang dipilih ada di tabel harga untuk dewasa dan anak-anak. **Langkah-langkah:** 1. Masukkan nama penumpang, alamat. 2. Pilih tujuan, kereta, gerbong, dan tanggal. 3. Atur spinner “Dewasa” ke 2 (atau angka > 0). 4. Atur spinner “Anak-anak” ke 1 (atau angka > 0). 5. Verifikasi harga “Total” dihitung dengan benar berdasarkan harga dewasa dan anak-anak. 6. Masukkan jumlah yang valid di “Bayar”. 7. Tekan Enter di kolom “Bayar”. 8. Klik “Simpan”. **Hasil yang Diharapkan:** - Pemesanan disimpan ke tabel penumpang. - Proses pembuatan tiket dimulai. - Jendela Beranda disegarkan/dibuka kembali. **Hasil Aktual:** Mirip dengan TC\_BOOK\_001. Metode dewasa() dan anak() dengan benar menggunakan nilai spinner dan harga dari label hdewasa/hanak. Metode simpan() menyisipkan nilai-nilai ini. JasperReports dipanggil. **Status:** Lulus

**ID:** TC\_BOOK\_003 **Judul:** Upaya pemesanan tanpa memilih tujuan. **Prakondisi:** - Pengguna sudah login dan berada di layar Beranda. **Langkah-langkah:** 1. Isi semua kolom wajib lainnya (nama, kereta, kelas, penumpang). 2. Biarkan “Stasiun Tujuan” pada keadaan default/tidak dipilih (jika berlaku, atau coba lanjutkan jika itu adalah pilihan wajib dari daftar). 3. Klik “Simpan”. **Hasil yang Diharapkan:** - Pesan kesalahan ditampilkan yang menunjukkan bahwa tujuan wajib diisi. - Pemesanan tidak disimpan. **Hasil Aktual:** Tidak ada validasi eksplisit di simpan(). stasiun\_tujuan.getSelectedItem() kemungkinan mengembalikan item pertama secara default. Kueri INSERT dilanjutkan. Waktu berangkat/tiba mungkin salah jika bergantung pada tujuan yang valid tertentu di ckeretaActionPerformed. Pemesanan disimpan dengan tujuan yang mungkin default/tidak diinginkan. **Status:** Gagal

**ID:** TC\_BOOK\_004 **Judul:** Upaya pemesanan tanpa memilih kereta. **Prakondisi:** - Pengguna sudah login dan berada di layar Beranda. **Langkah-langkah:** 1. Isi semua kolom wajib lainnya (nama, tujuan, kelas, penumpang). 2. Biarkan “Kereta Api” pada keadaan default/kosong (misalnya, item kosong pertama). 3. Klik “Simpan”. **Hasil yang Diharapkan:** - Pesan kesalahan ditampilkan yang menunjukkan bahwa pemilihan kereta wajib diisi, atau aplikasi menanganinya dengan baik (misalnya, tidak ada perhitungan harga). - Pemesanan tidak disimpan jika pemilihan wajib untuk penetapan harga/penyimpanan. **Hasil Aktual:** Tidak ada validasi eksplisit di simpan(). ckereta.getSelectedItem() mengembalikan "" (item kosong default). Kueri INSERT menyimpan "" untuk nama kereta. Label berangkat dan tiba tetap “00.00 WIB”. Perhitungan harga mungkin didasarkan pada hdewasa/hanak yang basi jika tidak ada kereta berarti tidak ada harga kelas yang dimuat. **Status:** Gagal

**ID:** TC\_BOOK\_005 **Judul:** Upaya pemesanan tanpa memilih kelas gerbong. **Prakondisi:** - Pengguna sudah login dan berada di layar Beranda. **Langkah-langkah:** 1. Isi semua kolom wajib lainnya (nama, tujuan, kereta, penumpang). 2. Biarkan “Gerbong” pada keadaan default/kosong. 3. Klik “Simpan”. **Hasil yang Diharapkan:** - Pesan kesalahan ditampilkan yang menunjukkan bahwa kelas gerbong wajib diisi untuk perhitungan harga. - Pemesanan tidak disimpan. **Hasil Aktual:** Tidak ada validasi eksplisit di simpan(). cgerbong.getSelectedItem() mengembalikan " " (item kosong default). kelasActionPerformed dipanggil, tetapi tidak ada kondisi yang cocok untuk " ", jadi hdewasa dan hanak tidak diperbarui (atau “0”). Perhitungan ltotal menggunakan harga yang salah/nol ini. Kueri INSERT menyimpan " " untuk gerbong dan total yang salah. **Status:** Gagal

**ID:** TC\_BOOK\_006 **Judul:** Upaya pemesanan tanpa memasukkan nama penumpang. **Prakondisi:** - Pengguna sudah login dan berada di layar Beranda. **Langkah-langkah:** 1. Isi semua kolom wajib lainnya (tujuan, kereta, kelas, penumpang). 2. Biarkan “Nama Penumpang” kosong. 3. Klik “Simpan”. **Hasil yang Diharapkan:** - Pesan kesalahan ditampilkan (misalnya, kesalahan database untuk kolom non-nullable, atau pesan validasi spesifik). - Pemesanan tidak disimpan. **Hasil Aktual:** Tidak ada validasi eksplisit di simpan(). fnama.getText() kosong. Kueri INSERT mencoba menyimpan "" untuk nama. Jika kolom DB nama di penumpang adalah NOT NULL dan tidak mengizinkan string kosong, eksekusi SQL gagal, dan JOptionPane generik menampilkan kesalahan SQL. Jika tidak, nama kosong disimpan. **Status:** Gagal

**ID:** TC\_BOOK\_007 **Judul:** Upaya pemesanan dengan nol dewasa dan nol anak-anak. **Prakondisi:** - Pengguna sudah login dan berada di layar Beranda. **Langkah-langkah:** 1. Isi semua kolom wajib lainnya (nama, tujuan, kereta, kelas). 2. Atur spinner “Dewasa” ke 0. 3. Atur spinner “Anak-anak” ke 0. 4. Klik “Simpan”. **Hasil yang Diharapkan:** - Pesan kesalahan ditampilkan yang menunjukkan bahwa setidaknya satu penumpang diperlukan, atau total harga nol dan penyimpanan dicegah. - Pemesanan tidak disimpan. **Hasil Aktual:** Tidak ada validasi eksplisit di simpan(). sdewasa.getValue() dan sanak.getValue() adalah 0. ltotal menjadi “Rp. 0”. Kueri INSERT menyimpan pemesanan dengan 0 dewasa, 0 anak-anak, dan total 0. **Status:** Gagal

**ID:** TC\_BOOK\_008 **Judul:** Verifikasi perhitungan harga dinamis berdasarkan kereta, kelas, dan jumlah penumpang. **Prakondisi:** - Pengguna sudah login dan berada di layar Beranda. - Harga yang diketahui ada di tabel harga untuk kelas dan status dewasa/anak yang berbeda. **Langkah-langkah:** 1. Pilih “Kereta Api”. 2. Pilih “Gerbong” (misalnya, “Eksekutif 1”). 3. Atur spinner “Dewasa” ke 1, “Anak-anak” ke 0. Catat “Total”. 4. Ubah spinner “Dewasa” ke 2. Verifikasi “Total” menjadi dua kali lipat. 5. Atur spinner “Anak-anak” ke 1. Verifikasi “Total” bertambah sebesar harga anak untuk kelas tersebut. 6. Ubah “Gerbong” ke kelas lain (misalnya, “Ekonomi 1”). 7. Verifikasi “Total” diperbarui berdasarkan harga kelas baru untuk jumlah dewasa dan anak-anak saat ini. **Hasil yang Diharapkan:** - Kolom “Total” diperbarui dengan benar dan dinamis saat kereta, kelas, dan jumlah penumpang berubah, mencerminkan harga dari tabel harga. - Label hdewasa dan hanak menampilkan harga per orang yang benar untuk kelas yang dipilih. **Hasil Aktual:** Pemilihan cgerbong memicu kelasActionPerformed yang menjalankan kueri ke tabel harga dan mengatur label hdewasa/hanak. Spinner sdewasa/sanak memicu metode dewasa()/anak() yang menggunakan nilai label ini untuk menghitung ltotal. Logika tampak benar. **Status:** Lulus

**ID:** TC\_BOOK\_009 **Judul:** Verifikasi proses pembuatan tiket dipanggil. **Prakondisi:** - Pemesanan berhasil dibuat (seperti pada TC\_BOOK\_001). **Langkah-langkah:** 1. Selesaikan semua langkah untuk pemesanan yang berhasil. 2. Klik “Simpan”. **Hasil yang Diharapkan:** - Metode JasperFillManager.fillReport dan JasperViewer.viewReport dipanggil dengan “src/report/Tiket.jasper” dan parameter ID penumpang yang benar. - (Konseptual) Tiket akan ditampilkan jika lingkungan tampilan tersedia. **Hasil Aktual:** Metode simpan(), setelah INSERT database yang berhasil, memanggil JasperFillManager.fillReport dengan “src/report/Tiket.jasper” dan JasperViewer.viewReport. ID penumpang (fid.getText()) diteruskan dengan benar sebagai parameter. **Status:** Lulus

## Skenario Panel Admin - Manajemen Pengguna

**ID:** TC\_ADMIN\_USER\_001 **Judul:** Lihat daftar pengguna admin. **Prakondisi:** - Pengguna sudah login dan berada di layar Panel Admin (Admin.java). - Pengguna admin ada di tabel admin. **Langkah-langkah:** 1. Navigasi ke Panel Admin. **Hasil yang Diharapkan:** - Tabel di bawah batas berjudul “Admin” diisi dengan data pengguna admin yang ada (No, Nama, Username, Password). **Hasil Aktual:** Konstruktor Admin() memanggil data(), yang menjalankan SELECT \* FROM admin order by id ASC dan mengisi JTable admin. **Status:** Lulus

**ID:** TC\_ADMIN\_USER\_002 **Judul:** Tambah pengguna admin baru dengan data valid. **Prakondisi:** - Pengguna berada di layar Panel Admin. **Langkah-langkah:** 1. Masukkan nama unik di kolom “Nama”. 2. Masukkan username unik di kolom “Username”. 3. Masukkan password di kolom “Password”. 4. Klik tombol “Simpan” di bawah bagian “Admin”. **Hasil yang Diharapkan:** - Pesan sukses “Berhasil” ditampilkan. - Pengguna admin baru muncul di tabel admin. - Kolom input dikosongkan. **Hasil Aktual:** btnadminActionPerformed di Admin.java menjalankan kueri INSERT INTO admin. Jika berhasil, ia menampilkan pesan “Berhasil”, memuat ulang tabel menggunakan data(), dan mengosongkan kolom input menggunakan bersih(). **Status:** Lulus

**ID:** TC\_ADMIN\_USER\_003 **Judul:** Upaya menambah pengguna admin dengan username yang sudah ada. **Prakondisi:** - Pengguna berada di layar Panel Admin. - Pengguna admin dengan username ‘testuser’ sudah ada. **Langkah-langkah:** 1. Masukkan nama apa pun di “Nama”. 2. Masukkan ‘testuser’ di “Username”. 3. Masukkan password apa pun di “Password”. 4. Klik “Simpan”. **Hasil yang Diharapkan:** - Pesan kesalahan ditampilkan (kemungkinan kesalahan pelanggaran batasan database untuk username unik). - Pengguna tidak ditambahkan ke tabel. **Hasil Aktual:** btnadminActionPerformed mencoba INSERT. Jika username sudah ada dan kolom username di tabel admin memiliki batasan UNIQUE, database akan menolak penyisipan. Blok catch(SQLException e) akan menampilkan JOptionPane yang berisi pesan kesalahan SQL (misalnya, “Duplicate entry…”). Tabel tidak akan menampilkan pengguna baru. **Status:** Lulus

**ID:** TC\_ADMIN\_USER\_004 **Judul:** Upaya menambah pengguna admin dengan username kosong. **Prakondisi:** - Pengguna berada di layar Panel Admin. **Langkah-langkah:** 1. Masukkan nama apa pun di “Nama”. 2. Biarkan “Username” kosong. 3. Masukkan password apa pun di “Password”. 4. Klik “Simpan”. **Hasil yang Diharapkan:** - Pesan kesalahan ditampilkan (kesalahan database untuk kolom non-nullable atau validasi spesifik). - Pengguna tidak ditambahkan. **Hasil Aktual:** btnadminActionPerformed mencoba INSERT dengan string kosong untuk username. Jika kolom username di tabel admin adalah NOT NULL dan tidak mengizinkan string kosong, database akan menolaknya. Blok catch(SQLException e) akan menampilkan kesalahan SQL. Pengguna tidak ditambahkan. **Status:** Lulus

**ID:** TC\_ADMIN\_USER\_005 **Judul:** Upaya menambah pengguna admin dengan password kosong. **Prakondisi:** - Pengguna berada di layar Panel Admin. **Langkah-langkah:** 1. Masukkan nama apa pun di “Nama”. 2. Masukkan username unik di “Username”. 3. Biarkan “Password” kosong. 4. Klik “Simpan”. **Hasil yang Diharapkan:** - Pengguna ditambahkan (password mungkin disimpan sebagai string kosong jika diizinkan oleh DB). Atau kesalahan jika password memiliki batasan. - *Koreksi mandiri: Aplikasi menyimpan password kosong tanpa kesalahan.* **Hasil Aktual:** btnadminActionPerformed mencoba INSERT dengan string kosong untuk password. Konfigurasi database standar untuk kolom password (VARCHAR) biasanya mengizinkan string kosong. Penyisipan berhasil, “Berhasil” ditampilkan, dan tabel diperbarui. **Status:** Lulus

**ID:** TC\_ADMIN\_USER\_006 **Judul:** Edit detail pengguna admin yang ada. **Prakondisi:** - Pengguna berada di layar Panel Admin. - Setidaknya satu pengguna admin ada. **Langkah-langkah:** 1. Klik pengguna admin yang ada di tabel. 2. Detail pengguna mengisi kolom “Nama”, “Username”, “Password”. ID disimpan di bayangan1. 3. Ubah kolom “Nama”. 4. Ubah kolom “Password”. 5. Klik tombol “Edit”. **Hasil yang Diharapkan:** - Pesan sukses “Berhasil” ditampilkan. - Tabel admin disegarkan dan menampilkan detail yang diperbarui untuk pengguna tersebut. - Kolom input dikosongkan. **Hasil Aktual:** adminMouseClicked mengisi kolom termasuk bayangan1 (ID). editActionPerformed menjalankan UPDATE admin ... WHERE id='...'. Jika berhasil, “Berhasil” ditampilkan, tabel dimuat ulang melalui data(), dan kolom dikosongkan melalui bersih(). **Status:** Lulus

**ID:** TC\_ADMIN\_USER\_007 **Judul:** Hapus pengguna admin. **Prakondisi:** - Pengguna berada di layar Panel Admin. - Setidaknya satu pengguna admin ada yang dapat dihapus. **Langkah-langkah:** 1. Klik pengguna admin yang ada di tabel untuk memilihnya. 2. Klik tombol “Hapus”. **Hasil yang Diharapkan:** - Pesan sukses “Berhasil” ditampilkan. - Tabel admin disegarkan dan pengguna yang dipilih tidak lagi terdaftar. - Kolom input dikosongkan. **Hasil Aktual:** adminMouseClicked mengisi bayangan1 (ID). hapusActionPerformed menjalankan DELETE FROM admin WHERE id='...'. Jika berhasil, “Berhasil” ditampilkan, tabel dimuat ulang melalui data(), dan kolom dikosongkan melalui bersih(). **Status:** Lulus

## Skenario Panel Admin - Manajemen Harga

**ID:** TC\_ADMIN\_PRICE\_001 **Judul:** Lihat daftar harga. **Prakondisi:** - Pengguna berada di layar Panel Admin. - Data harga ada di tabel harga. **Langkah-langkah:** 1. Navigasi ke Panel Admin. **Hasil yang Diharapkan:** - Tabel di bawah batas berjudul “Harga” diisi dengan data harga yang ada (No, Status, Kelas, Harga). **Hasil Aktual:** Konstruktor Admin() memanggil data\_harga(), yang menjalankan SELECT \* FROM harga dan mengisi JTable tableharga. **Status:** Lulus

**ID:** TC\_ADMIN\_PRICE\_002 **Judul:** Edit harga yang ada untuk kelas dan status tertentu. **Prakondisi:** - Pengguna berada di layar Panel Admin. - Data harga ada. **Langkah-langkah:** 1. Klik entri harga di tabel “Harga”. 2. Kolom “Status”, “Kelas”, dan “Harga” diisi. ID disimpan di bayangan1. 3. Ubah nilai di kolom “Harga” menjadi nilai numerik baru yang valid. 4. Klik tombol “Simpan” di bawah bagian “Harga”. **Hasil yang Diharapkan:** - Pesan sukses “Berhasil” ditampilkan. - Tabel “Harga” disegarkan dan menampilkan harga yang diperbarui untuk entri tersebut. - Kolom input dikosongkan. **Hasil Aktual:** tablehargaMouseClicked mengisi kolom dari baris yang dipilih, menyimpan ID di bayangan1. Pengguna mengedit kolom harga. Tombol “Simpan” di panel “Harga” (simpanActionPerformed) menjalankan UPDATE harga set harga='<nilai\_baru>' where id='<id\_dari\_bayangan1>'. Jika berhasil, “Berhasil” ditampilkan, data\_harga() memuat ulang tabel, dan bersih() mengosongkan kolom. **Status:** Lulus

**ID:** TC\_ADMIN\_PRICE\_003 **Judul:** Upaya menyimpan harga dengan nilai non-numerik. **Prakondisi:** - Pengguna berada di layar Panel Admin. - Data harga ada. **Langkah-langkah:** 1. Pilih entri harga dari tabel. 2. Masukkan “abc” (atau string non-numerik lainnya) di kolom “Harga”. 3. Klik “Simpan” di bawah bagian “Harga”. **Hasil yang Diharapkan:** - Pesan kesalahan ditampilkan (kemungkinan NumberFormatException atau kesalahan database jika mencoba menyimpan). - Harga tidak diperbarui. **Hasil Aktual:** Pengguna memasukkan “abc” ke dalam kolom harga. simpanActionPerformed menjalankan UPDATE harga set harga='abc' .... Jika kolom harga di database adalah numerik, ini akan menyebabkan kesalahan SQL. Blok catch(SQLException e) menampilkan JOptionPane dengan pesan kesalahan database. Harga tidak diperbarui. **Status:** Lulus

**ID:** TC\_ADMIN\_PRICE\_004 **Judul:** Upaya menyimpan harga dengan nilai negatif. **Prakondisi:** - Pengguna berada di layar Panel Admin. - Data harga ada. **Langkah-langkah:** 1. Pilih entri harga dari tabel. 2. Masukkan “-100” di kolom “Harga”. 3. Klik “Simpan” di bawah bagian “Harga”. **Hasil yang Diharapkan:** - Harga diperbarui menjadi -100 (jika tidak ada validasi yang mencegahnya). - *Koreksi mandiri: Aplikasi mengizinkan harga negatif.* **Hasil Aktual:** Pengguna memasukkan “-100” ke dalam kolom harga. simpanActionPerformed menjalankan UPDATE harga set harga='-100' .... Dengan asumsi kolom harga di database adalah tipe numerik bertanda, pembaruan berhasil. “Berhasil” ditampilkan, tabel dimuat ulang. Tidak ada validasi tingkat aplikasi yang mencegah harga negatif. **Status:** Lulus

## Skenario Manajemen Data Penumpang (Layar Penumpang)

**ID:** TC\_PASSENGER\_001 **Judul:** Lihat daftar penumpang yang telah memesan. **Prakondisi:** - Pengguna sudah login. - Pemesanan penumpang ada di tabel penumpang. **Langkah-langkah:** 1. Navigasi ke layar “Data Penumpang” dari menu. **Hasil yang Diharapkan:** - Tabel di layar “Data Penumpang” diisi dengan semua detail pemesanan penumpang yang ada. **Hasil Aktual:** Konstruktor Penumpang() memanggil metode data(), yang menjalankan SELECT \* FROM penumpang order by id ASC dan mengisi JTable penumpang dengan hasilnya. **Status:** Lulus

**ID:** TC\_PASSENGER\_002 **Judul:** Verifikasi bahwa tiket yang baru dipesan muncul di daftar penumpang. **Prakondisi:** - Pengguna sudah login. **Langkah-langkah:** 1. Catat jumlah penumpang saat ini di layar “Data Penumpang” (atau jika kosong). 2. Navigasi ke layar Beranda dan berhasil memesan tiket baru (misalnya, TC\_BOOK\_001). 3. Setelah memesan, navigasi kembali ke layar “Data Penumpang”. **Hasil yang Diharapkan:** - Detail tiket yang baru dipesan muncul sebagai baris baru di tabel penumpang. - Jumlah penumpang bertambah satu. **Hasil Aktual:** Setelah pemesanan berhasil di Home.java (yang menyisipkan ke tabel penumpang), menavigasi ke layar Penumpang (yang membuat instance Penumpang baru) akan memanggil data(), mengisi ulang tabel dengan data terbaru, termasuk pemesanan baru. **Status:** Lulus

**ID:** TC\_PASSENGER\_003 **Judul:** Uji fungsionalitas “Cetak” untuk data penumpang. **Prakondisi:** - Pengguna berada di layar “Data Penumpang”. **Langkah-langkah:** 1. Klik tombol “Cetak”. **Hasil yang Diharapkan:** - Metode JasperFillManager.fillReport dan JasperViewer.viewReport dipanggil dengan “src/report/penumpang.jasper”. - (Konseptual) Laporan yang mencantumkan semua penumpang akan ditampilkan jika lingkungan tampilan tersedia. **Hasil Aktual:** Listener aksi tombol “Cetak” (jButton1ActionPerformed) memanggil JasperFillManager.fillReport dengan “src/report/penumpang.jasper” dan JasperViewer.viewReport. Ini dengan benar memanggil proses pelaporan. **Status:** Lulus

**ID:** TC\_PASSENGER\_004 **Judul:** Uji fungsionalitas “Hapus Semua Data”. **Prakondisi:** - Pengguna berada di layar “Data Penumpang”. - Beberapa data penumpang ada. **Langkah-langkah:** 1. Klik tombol “Hapus” (Hapus Semua Data). 2. Dialog konfirmasi “Yakin Ingin Menghapus Data?” muncul. 3. Klik “Ya” (atau opsi afirmatif yang setara). **Hasil yang Diharapkan:** - Pesan sukses “Berhasil” ditampilkan. - Tabel penumpang menjadi kosong. - Semua catatan dihapus dari tabel penumpang di database. **Hasil Aktual:** Tombol “Hapus” (HapusActionPerformed) menampilkan dialog konfirmasi JOptionPane. Jika “Ya” dipilih, ia menjalankan DELETE FROM penumpang. Setelah penghapusan berhasil, “Berhasil” ditampilkan, dan data() dipanggil untuk menyegarkan tabel (yang sekarang akan kosong). **Status:** Lulus

**ID:** TC\_PASSENGER\_005 **Judul:** Uji fungsionalitas “Hapus Semua Data” - Batal. **Prakondisi:** - Pengguna berada di layar “Data Penumpang”. - Beberapa data penumpang ada. **Langkah-langkah:** 1. Klik tombol “Hapus” (Hapus Semua Data). 2. Dialog konfirmasi “Yakin Ingin Menghapus Data?” muncul. 3. Klik “Tidak” (atau opsi negatif yang setara). **Hasil yang Diharapkan:** - Tidak ada pesan sukses mengenai penghapusan yang ditampilkan. - Tabel penumpang tetap tidak berubah. - Tidak ada catatan yang dihapus dari tabel penumpang. **Hasil Aktual:** Tombol “Hapus” (HapusActionPerformed) menampilkan dialog konfirmasi JOptionPane. Jika “Tidak” dipilih, kondisi if (pesan == JOptionPane.YES\_OPTION) bernilai false, dan tidak ada operasi database atau penyegaran tabel yang terjadi selain pemuatan awal. **Status:** Lulus

# Laporan Pengujian: Aplikasi 1TiketKeretaApi

## 1. Pendahuluan

**Tujuan Pengujian:** Tujuan dari pengujian ini adalah untuk mengevaluasi fungsionalitas inti dari aplikasi Java Swing “1TiketKeretaApi” melalui analisis kode statis dan simulasi eksekusi skenario pengujian. Laporan ini mendokumentasikan temuan-temuan, termasuk bug yang teridentifikasi, area untuk peningkatan, dan penilaian keseluruhan terhadap perilaku aplikasi berdasarkan kode sumbernya.

**Metodologi:** Karena keterbatasan dalam menjalankan aplikasi GUI secara langsung dan dependensinya terhadap database di lingkungan saat ini, pengujian dilakukan melalui:

1. **Analisis Kode Statis:** Meninjau kode sumber Java untuk setiap modul fungsional (Login.java, Home.java, Admin.java, Penumpang.java, Koneksi.java).
2. **Simulasi Skenario Pengujian:** Secara mental menjalankan skenario pengujian yang telah ditentukan (dari test\_scenarios.md) dengan menelusuri alur kode, memprediksi perilaku antarmuka pengguna (pesan dialog, navigasi), dan menganalisis interaksi database (kueri SQL).
3. **Dokumentasi:** Mencatat “Hasil Aktual” dan “Status” untuk setiap kasus uji berdasarkan analisis ini.

## 2. Gambaran Umum Aplikasi

Aplikasi “1TiketKeretaApi” adalah aplikasi desktop Java Swing yang dirancang sebagai sistem pemesanan tiket kereta api. Aplikasi ini menyediakan fungsionalitas bagi pengguna (yang diasumsikan sebagai administrator setelah login) untuk memesan tiket bagi penumpang, mengelola pengguna administratif, dan mengatur harga tiket.

Fungsionalitas utama meliputi:

* **Login:** Mengautentikasi pengguna terhadap tabel admin di database MySQL.
* **Pemesanan Tiket (Home.java):** Memungkinkan pemesanan tiket dengan memasukkan detail penumpang (ID, nama, alamat), memilih stasiun awal/tujuan, kereta, kelas gerbong, tanggal, dan jumlah penumpang dewasa/anak-anak. Aplikasi ini secara dinamis menghitung harga dan dimaksudkan untuk menghasilkan tiket menggunakan JasperReports.
* **Panel Admin (Admin.java):**
  + **Manajemen Pengguna:** Menambah, melihat, mengedit, dan menghapus pengguna admin (data disimpan di tabel admin).
  + **Manajemen Harga:** Melihat dan mengedit harga tiket untuk status yang berbeda (dewasa/anak) dan kelas gerbong (data disimpan di tabel harga).
* **Manajemen Data Penumpang (Penumpang.java):** Melihat daftar semua penumpang yang telah memesan tiket, mencetak laporan penumpang (JasperReports), dan menghapus semua catatan penumpang dari tabel penumpang.

## 3. Lingkungan Pengujian

* **Artefak:** File kode sumber Java untuk aplikasi (Login.java, Home.java, Admin.java, Penumpang.java, Koneksi.java, Index.java).
* **Skema Database (Disimpulkan):**
  + Tabel admin (untuk login)
  + Tabel penumpang (untuk menyimpan pemesanan)
  + Tabel harga (untuk menyimpan informasi harga)
* **Metodologi:** Tinjauan kode manual, analisis statis, dan simulasi eksekusi skenario pengujian. Tidak ada eksekusi langsung atau interaksi database yang dilakukan.

## 4. Lingkup Pengujian

Fungsionalitas berikut dicakup selama pengujian:

* Autentikasi login.
* Proses Pemesanan Tiket, termasuk kolom input, perhitungan dinamis, dan operasi penyimpanan.
* Panel Admin:
  + Manajemen pengguna admin (operasi CRUD).
  + Manajemen harga tiket (lihat dan perbarui).
* Manajemen Data Penumpang:
  + Melihat daftar penumpang yang dipesan.
  + Verifikasi pemesanan baru muncul.
  + Inisiasi cetak laporan.
  + Penghapusan semua data penumpang.

Pembuatan JasperReport diuji secara konseptual dengan memverifikasi apakah alur kode untuk memanggil pembuatan laporan diimplementasikan dengan benar.

## 5. Ringkasan Pelaksanaan Pengujian

Sebanyak 26 skenario pengujian dianalisis.

* **Total Pengujian:** 26
* **Pengujian Lulus:** 21
* **Pengujian Gagal:** 5

**Tabel Skenario Pengujian:**

| ID | Judul | Status | Hasil Aktual Singkat / Catatan |
| --- | --- | --- | --- |
| TC\_LOGIN\_001 | Login berhasil dengan kredensial admin yang valid. | Lulus | Alur kode untuk login berhasil dan navigasi ke layar Beranda sudah benar. |
| TC\_LOGIN\_002 | Login gagal dengan username tidak valid. | Lulus | Menampilkan pesan “Gagal Login Oi” seperti yang diharapkan. |
| TC\_LOGIN\_003 | Login gagal dengan password tidak valid. | Lulus | Menampilkan pesan “Gagal Login Oi” seperti yang diharapkan. |
| TC\_LOGIN\_004 | Login gagal dengan username kosong. | Lulus | Menampilkan pesan “Gagal Login Oi”. |
| TC\_LOGIN\_005 | Login gagal dengan password kosong. | Lulus | Menampilkan pesan “Gagal Login Oi”. |
| TC\_BOOK\_001 | Pemesanan tiket berhasil dengan semua input valid untuk satu orang dewasa. | Lulus | Alur kode untuk menyimpan pemesanan dan memanggil JasperReports sudah benar. |
| TC\_BOOK\_002 | Pemesanan tiket berhasil untuk beberapa dewasa dan anak-anak. | Lulus | Perhitungan harga dan logika penyimpanan untuk beberapa penumpang sudah benar. |
| TC\_BOOK\_003 | Upaya pemesanan tanpa memilih tujuan. | Gagal | Tidak ada validasi; menyimpan dengan tujuan default/kosong. |
| TC\_BOOK\_004 | Upaya pemesanan tanpa memilih kereta. | Gagal | Tidak ada validasi; menyimpan nama kereta kosong, waktu berangkat/tiba salah. |
| TC\_BOOK\_005 | Upaya pemesanan tanpa memilih kelas gerbong. | Gagal | Tidak ada validasi; menyimpan gerbong kosong, menggunakan harga salah/nol. |
| TC\_BOOK\_006 | Upaya pemesanan tanpa memasukkan nama penumpang. | Gagal | Tidak ada validasi; bergantung pada batasan DB (jika ada) untuk mencegah nama kosong. |
| TC\_BOOK\_007 | Upaya pemesanan dengan nol dewasa dan nol anak-anak. | Gagal | Tidak ada validasi; mengizinkan pemesanan untuk nol penumpang dengan total harga nol. |
| TC\_BOOK\_008 | Verifikasi perhitungan harga dinamis. | Lulus | Logika perhitungan harga berdasarkan kelas dan jumlah penumpang tampak benar. |
| TC\_BOOK\_009 | Verifikasi proses pembuatan tiket dipanggil. | Lulus | Alur kode pemanggilan JasperReports sudah benar. |
| TC\_ADMIN\_USER\_001 | Lihat daftar pengguna admin. | Lulus | Metode data() memuat admin ke tabel dengan benar. |
| TC\_ADMIN\_USER\_002 | Tambah pengguna admin baru dengan data valid. | Lulus | btnadminActionPerformed menyisipkan admin baru dengan benar. |
| TC\_ADMIN\_USER\_003 | Upaya menambah admin dengan username yang sudah ada. | Lulus | Gagal di tingkat DB (pelanggaran batasan unik), pesan kesalahan ditampilkan. |
| TC\_ADMIN\_USER\_004 | Upaya menambah admin dengan username kosong. | Lulus | Gagal di tingkat DB (jika ada batasan NOT NULL), pesan kesalahan ditampilkan. |
| TC\_ADMIN\_USER\_005 | Upaya menambah admin dengan password kosong. | Lulus | Aplikasi mengizinkan penyimpanan admin dengan password kosong. |
| TC\_ADMIN\_USER\_006 | Edit detail pengguna admin yang ada. | Lulus | editActionPerformed memperbarui detail admin dengan benar. |
| TC\_ADMIN\_USER\_007 | Hapus pengguna admin. | Lulus | hapusActionPerformed menghapus admin dengan benar. |
| TC\_ADMIN\_PRICE\_001 | Lihat daftar harga. | Lulus | Metode data\_harga() memuat harga ke tabel dengan benar. |
| TC\_ADMIN\_PRICE\_002 | Edit harga yang ada untuk kelas dan status tertentu. | Lulus | simpanActionPerformed (panel harga) memperbarui harga dengan benar. |
| TC\_ADMIN\_PRICE\_003 | Upaya menyimpan harga dengan nilai non-numerik. | Lulus | Gagal di tingkat DB (jika kolom numerik), pesan kesalahan ditampilkan. |
| TC\_ADMIN\_PRICE\_004 | Upaya menyimpan harga dengan nilai negatif. | Lulus | Aplikasi mengizinkan penyimpanan harga negatif. |
| TC\_PASSENGER\_001 | Lihat daftar penumpang yang telah memesan. | Lulus | Metode data() di Penumpang.java memuat data penumpang dengan benar. |
| TC\_PASSENGER\_002 | Verifikasi bahwa tiket yang baru dipesan muncul di daftar penumpang. | Lulus | Penyisipan pemesanan baru di Home dan pemuatan ulang di Penumpang berfungsi. |
| TC\_PASSENGER\_003 | Uji fungsionalitas “Cetak” untuk data penumpang. | Lulus | Pemanggilan JasperReports untuk daftar penumpang sudah benar. |
| TC\_PASSENGER\_004 | Uji fungsionalitas “Hapus Semua Data”. | Lulus | HapusActionPerformed menghapus semua penumpang dengan benar setelah konfirmasi. |
| TC\_PASSENGER\_005 | Uji fungsionalitas “Hapus Semua Data” - Batal. | Lulus | Pembatalan “Hapus Semua Data” berfungsi seperti yang diharapkan. |

## 6. Temuan Utama dan Isu

### 6.1. Bug/Kasus Uji Gagal

Kasus uji berikut gagal, terutama karena kurangnya validasi input dalam modul Pemesanan Tiket (Home.java):

* **TC\_BOOK\_003: Upaya pemesanan tanpa memilih tujuan.**
  + **Isu:** Aplikasi tidak memvalidasi apakah stasiun tujuan telah dipilih. Aplikasi melanjutkan untuk menyimpan pemesanan, kemungkinan dengan item default (pertama) di JComboBox, atau nilai kosong jika ComboBox mengizinkannya. Hal ini dapat menyebabkan data pemesanan yang salah dan memengaruhi logika dependen seperti penentuan waktu berangkat/tiba.
* **TC\_BOOK\_004: Upaya pemesanan tanpa memilih kereta.**
  + **Isu:** Aplikasi mengizinkan pemesanan dengan string kosong ("") sebagai nama kereta karena JComboBox untuk kereta memiliki item kosong awal dan tidak ada validasi. Hal ini menghasilkan data yang tidak berarti untuk kereta, dan waktu berangkat/tiba tidak diatur (tetap “00.00 WIB”).
* **TC\_BOOK\_005: Upaya pemesanan tanpa memilih kelas gerbong.**
  + **Isu:** Mirip dengan pemilihan kereta, kelas gerbong kosong (" ") dapat disimpan karena kurangnya validasi. Hal ini menyebabkan perhitungan harga yang salah, karena logika untuk mengambil harga (kelasActionPerformed) tidak akan menemukan kriteria yang cocok, sehingga harga menjadi “0” atau basi dari pilihan sebelumnya.
* **TC\_BOOK\_006: Upaya pemesanan tanpa memasukkan nama penumpang.**
  + **Isu:** Aplikasi tidak melakukan validasi sisi klien untuk memastikan nama penumpang dimasukkan. Aplikasi mencoba menyimpan string kosong. Ini hanya akan gagal jika kolom database yang sesuai (nama di tabel penumpang) memiliki batasan NOT NULL yang juga mencegah string kosong. Jika tidak, pemesanan tanpa nama penumpang akan dibuat.
* **TC\_BOOK\_007: Upaya pemesanan dengan nol dewasa dan nol anak-anak.**
  + **Isu:** Sistem mengizinkan pemesanan dilakukan meskipun jumlah dewasa dan anak-anak adalah nol. Hal ini menghasilkan pemesanan dengan total harga nol dan tidak ada penumpang aktual, yang secara logis salah.

### 6.2. Area Peningkatan/Observasi

Selain kegagalan pengujian langsung, analisis statis mengungkapkan beberapa area di mana aplikasi dapat ditingkatkan untuk ketahanan dan kemudahan pemeliharaan:

* **Kurangnya Validasi Input Secara Umum (Pemesanan Tiket):** Seperti yang disorot oleh kasus uji yang gagal, modul pemesanan tiket (Home.java) sangat kurang dalam validasi input proaktif untuk kolom-kolom wajib (tujuan, kereta, kelas, nama penumpang) dan minimum logis (setidaknya satu penumpang). Bergantung semata-mata pada batasan database (jika ada) tidak ramah pengguna.
* **Harga Negatif Diizinkan (Panel Admin - Manajemen Harga):** (TC\_ADMIN\_PRICE\_004) Sistem mengizinkan nilai negatif dimasukkan dan disimpan untuk harga tiket. Hal ini kemungkinan tidak diinginkan dan harus dicegah dengan validasi.
* **Pembuatan ID Penumpang (autoid() di Home.java):** Metode saat ini SELECT COUNT(id) as kode FROM penumpang kemudian menambahkan 1 ke hitungan (misalnya, “MR.X”) tidak tangguh. Ini dapat menyebabkan ID duplikat jika catatan dihapus atau dalam skenario akses bersamaan (meskipun kurang mungkin untuk jenis aplikasi desktop ini). Pendekatan yang lebih tangguh adalah menggunakan kunci primer auto-increment database secara langsung, atau UUID.
* **Penanganan Tanggal untuk Database (simpan() di Home.java):** Kode menggunakan tanggal.getDate() dan secara langsung menggabungkannya ke dalam string SQL. Cara java.util.Date.toString() memformat tanggal mungkin tidak selalu kompatibel dengan format tanggal yang diharapkan MySQL, yang berpotensi menyebabkan kesalahan atau penyimpanan tanggal yang salah tergantung pada lokal atau konfigurasi MySQL. Menggunakan PreparedStatement dengan setDate() adalah praktik yang direkomendasikan.
* **Password Admin Kosong Diizinkan (Panel Admin - Manajemen Pengguna):** (TC\_ADMIN\_USER\_005) Aplikasi mengizinkan pembuatan pengguna admin dengan password kosong. Meskipun pengujian lulus karena sistem berfungsi sesuai kode, ini adalah masalah keamanan dan idealnya harus dilarang melalui validasi.
* **Penanganan Kesalahan:** Blok catch(Exception e) atau catch(SQLException e) generik digunakan, menampilkan JOptionPane.showMessageDialog(null,e);. Meskipun ini menampilkan kesalahan, ini tidak terlalu ramah pengguna karena sering kali mengekspos detail SQL mentah atau pengecualian Java. Pesan kesalahan yang lebih spesifik akan lebih baik.
* **Struktur Kode di Home.java (ckeretaActionPerformed):** Rantai if-else-if yang panjang untuk mengatur waktu berangkat/tiba berdasarkan kereta dan tujuan sulit dipelihara dan rentan terhadap kesalahan. Ini dapat direfaktor, mungkin menggunakan struktur data atau pendekatan yang lebih berorientasi objek jika jadwal kereta menjadi kompleks.

## 7. Penilaian Keseluruhan

Aplikasi “1TiketKeretaApi” menyediakan kerangka kerja dasar untuk fungsionalitas yang dimaksudkan. Bagian Admin dan Data Penumpang tampak relatif stabil berdasarkan alur kode untuk operasi inti mereka (CRUD untuk admin/harga, melihat/menghapus/melaporkan untuk penumpang).

Namun, **modul Pemesanan Tiket (Home.java) memerlukan perhatian signifikan**. Kurangnya validasi input mendasar untuk kolom-kolom kritis (tujuan, kereta, kelas, nama penumpang, jumlah penumpang) menyebabkan beberapa kasus uji gagal dan dapat mengakibatkan data yang tidak konsisten atau tidak valid dalam database, serta pengalaman pengguna yang buruk.

Rekomendasi untuk peningkatan meliputi:

* Mengimplementasikan validasi input yang komprehensif dalam modul Home.java sebelum mencoba menyimpan data pemesanan.
* Menambahkan validasi untuk mencegah harga negatif.
* Merefaktor pembuatan ID penumpang dan penanganan tanggal untuk operasi database guna meningkatkan ketahanan.
* Meningkatkan penanganan kesalahan untuk memberikan pesan yang lebih ramah pengguna.
* Meninjau kebijakan mengenai password admin kosong.

Mengatasi isu-isu ini, terutama validasi input dalam pemesanan tiket, akan secara substansial meningkatkan keandalan dan kegunaan aplikasi.