LAPORAN PRAKTIKUM

BAHASA PEMROGRAMAN 2

MODUL 8

Dosen pengampu : Yulyanto, S.Kom., M.TI.

Disusun oleh :

Nama : Muhammad Rizal Nurfirdaus

NIM : 20230810088

Jadwal : Rabu, 14:40 – 16:15

Kelas : TINFC-2023-04

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS KUNINGAN

2025

DAFTAR ISI

BAB I

PRETEST

1. Tuliskan Library yang dibutuhkan untuk membuat report di netbeans.

Kerjakan pda selembar kertas dengan waktu 5 menit dan dikumpulkan melalui Asisten lab.

Daftar Library JasperReports

a. jasperreports-6.21.0.jar

b. commons-logging-1.2.jar

c. commons-digester-2.1.jar

d. commons-collections-3.2.2.jar

e. itext-2.1.7.js2.jar (versi ini kompatibel dengan JasperReports)

f. poi-5.2.5.jar

g. poi-ooxml-5.2.5.jar

h. poi-ooxml-schemas-4.1.2.jar

i. xmlbeans-5.1.1.jar

j. groovy-all-2.4.21.jar

k. jfreechart-1.5.4.jar

l. barbecue-1.5-beta1.jar

m. commons-beanutils-1.9.4.jar

n. commons-lang-2.6.jar

o. commons-javaflow-20060411.jar

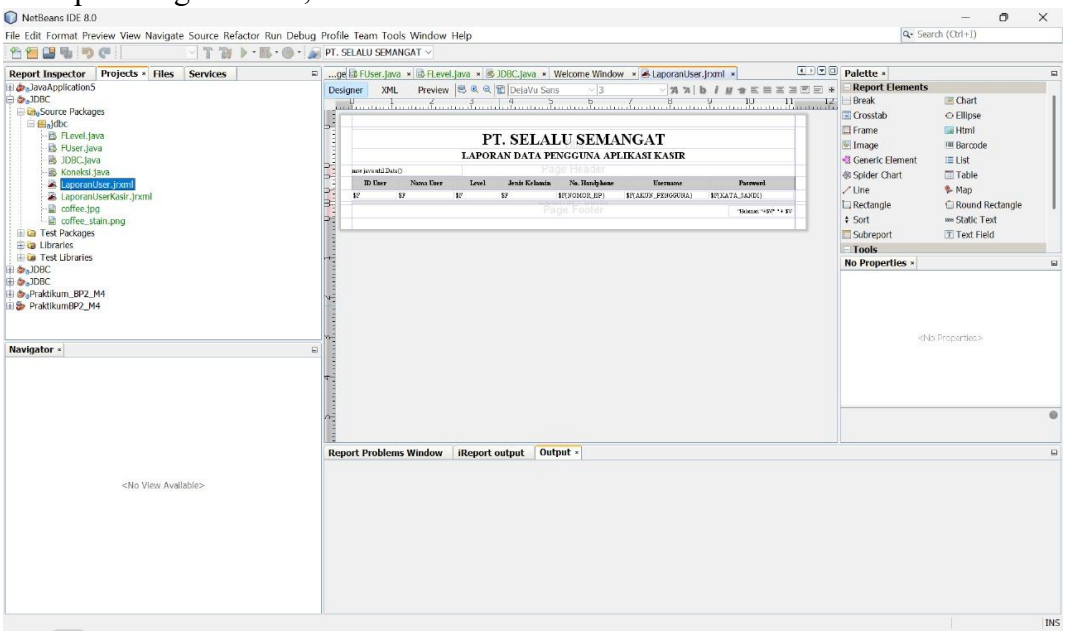
BAB II

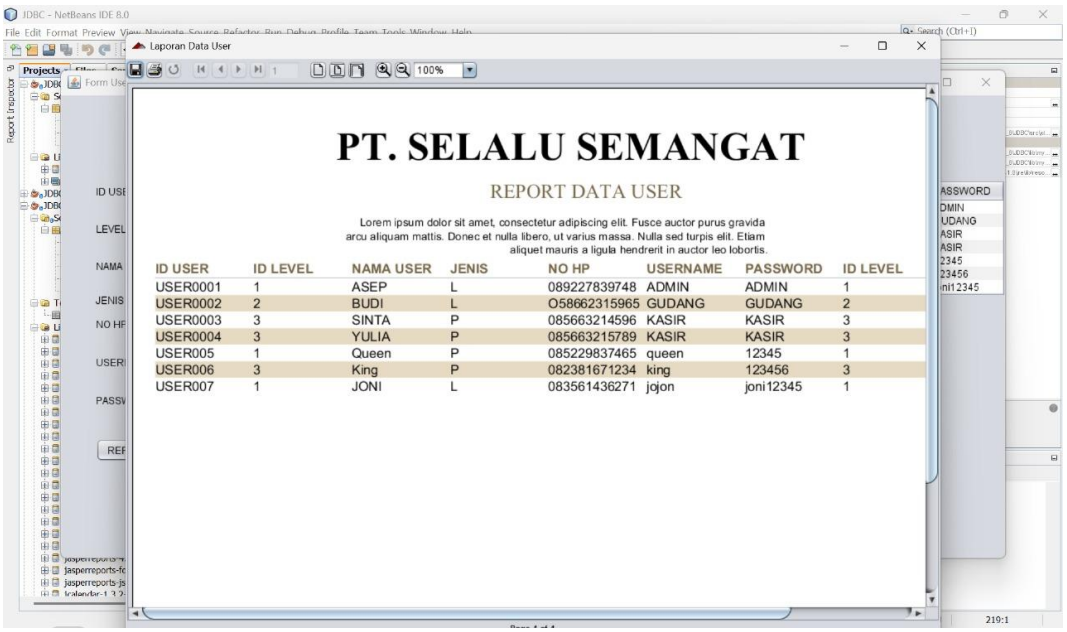
PRAKTIKUM

1. Buatkan report untuk FUser

```
import java.io.File; // Perbaikan import
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.util.HashMap;
import java.util.Map;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.engine.design.JasperDesign;
import net.sf.jasperreports.engine.xml.JRXmlLoader;
import net.sf.jasperreports.view.JasperViewer;


JasperReport JasRep;
   JasperPrint JasPri;
   Map<String, Object> param = new HashMap<>(); // Menggunakan generic type
```

JasperDesign JasDes;





Analisis : Kode yang ditampilkan merupakan bagian dari proses pembuatan dan penampilan laporan menggunakan JasperReports di Java melalui NetBeans. Proses ini dimulai dengan memuat file desain laporan berformat .jrxml menggunakan JRXmlLoader, lalu mengompilasinya menjadi .jasper melalui JasperCompileManager. Setelah itu, laporan diisi dengan data dari database menggunakan JasperFillManager, dan hasilnya ditampilkan ke pengguna menggunakan JasperViewer. Selain itu, Map<String, Object> digunakan untuk menyisipkan parameter dinamis ke dalam laporan jika diperlukan, seperti filter tanggal atau nama pengguna.

Agar laporan dapat ditampilkan dengan sukses, pastikan koneksi database aktif, file .jrxml berada di path yang benar, dan semua library JasperReports sudah

ditambahkan ke project. Struktur ini sangat berguna untuk membuat laporan dinamis berbasis data, seperti laporan pemesanan, pengguna, atau jadwal dalam aplikasi desktop berbasis Java.

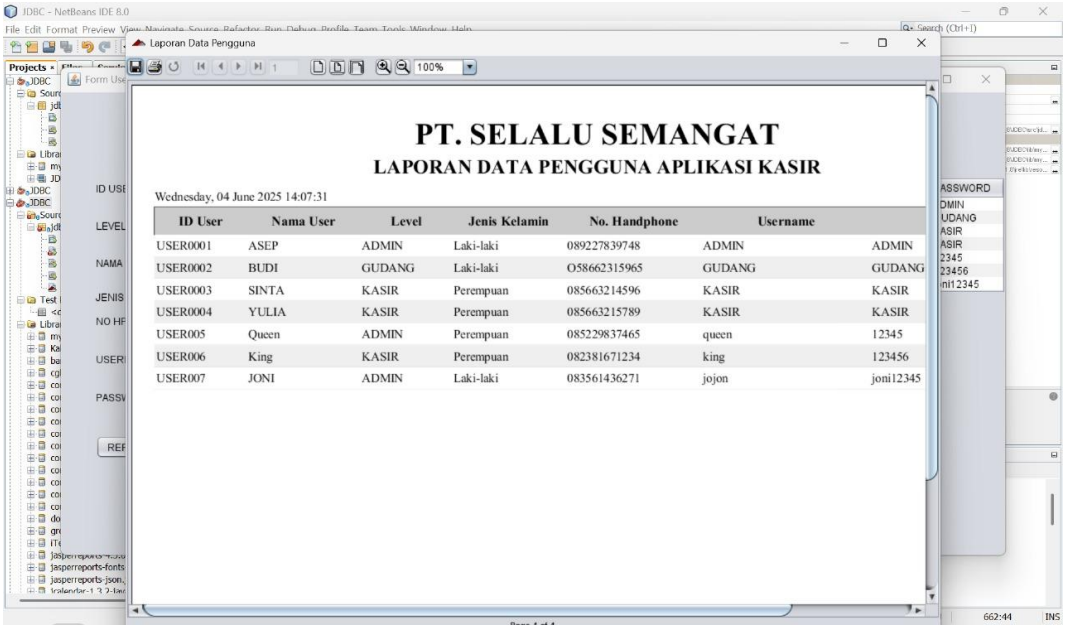
BAB III

POSTTEST

1. Modifikasi tampilan report dan query yang lebih spesifik

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    if (koneksi.con == null) {
        JOptionPane.showMessageDialog(this, "Koneksi Database Gagal! Tidak dapat mencetak report.", "Error Koneksi", JOptionPane.ERROR_MESSAGE);
        return;
    }
    try {
        // Pastikan path ke file .jrxml benar relatif terhadap root proyek atau classpath.
        // Jika file ada di src/jdbc/LaporanUser.jrxml:
        File fileReport = new File("src/jdbc/LaporanUser.jrxml"); // Menggunakan nama variabel yang lebih deskriptif

        if (!fileReport.exists()) {
            JOptionPane.showMessageDialog(this, "File laporan LaporanUser.jrxml tidak ditemukan di src/jdbc/", "Error File Laporan", JOptionPane.ERROR_MESSAGE);
            // Anda bisa juga mencoba mencari di root classpath jika struktur proyek berbeda
            // InputStream reportStream = getClass().getResourceAsStream("/jdbc/LaporanUser.jrxml");
            // if (reportStream == null) {
            //     JOptionPane.showMessageDialog(this, "File laporan LaporanUser.jrxml tidak ditemukan.", "Error File", JOptionPane.ERROR_MESSAGE);
            //     return;
            // }
            // JasDes = JRXmlLoader.load(reportStream);
            return;
```

```
            }

            JasDes = JRXmlLoader.load(fileReport.getAbsolutePath()); // Menggunakan path absolut lebih aman untuk File
            param.clear(); // Bersihkan parameter jika ada penggunaan sebelumnya

            JasRep = JasperCompileManager.compileReport(JasDes);
            JasPri = JasperFillManager.fillReport(JasRep, param, koneksi.con); // Menggunakan koneksi.con

            // Menampilkan laporan
            // JasperViewer.viewReport(JasPri, false); // false agar aplikasi tidak exit saat viewer ditutup

            // Alternatif jika JasperViewer.viewReport menyebabkan masalah atau untuk kontrol lebih
            JasperViewer viewer = new JasperViewer(JasPri, false); // false agar aplikasi tidak exit
            viewer.setTitle("Laporan Data User"); // Memberi judul pada window viewer
            viewer.setVisible(true);

        } catch (Exception e) { // Tangkap exception umum lainnya
            JOptionPane.showMessageDialog(this, "Terjadi kesalahan umum saat membuat laporan: " + e.getMessage(), "Error Umum", JOptionPane.ERROR_MESSAGE);
            e.printStackTrace(); // Penting untuk melihat detail error di konsol
        }
    }
```



Analisis : Kode dan tampilan laporan pada gambar menunjukkan keberhasilan penggunaan JasperReports untuk menampilkan data pengguna aplikasi kasir dalam format laporan yang rapi dan profesional. Di dalam metode jButton4ActionPerformed, proses pembuatan laporan dimulai dengan pengecekan koneksi ke database, lalu memuat file desain laporan .jrxml dari direktori src/jdbc/. Setelah file ditemukan dan dipastikan ada, file tersebut dikompilasi, diisi dengan data dari database menggunakan koneksi aktif (koneksi.con), dan ditampilkan menggunakan JasperViewer dengan judul khusus. Tampilan laporan pada gambar memperlihatkan data pengguna seperti ID, nama, level, jenis kelamin, nomor HP, dan username, dengan format tabel yang sudah diatur rapi dan mencantumkan judul laporan “PT. SELALU SEMANGAT – LAPORAN DATA PENGGUNA APLIKASI KASIR”.

Struktur kode sudah sangat baik karena menangani berbagai kemungkinan error seperti koneksi gagal dan file laporan tidak ditemukan, dengan memberikan umpan balik ke pengguna melalui JOptionPane. Penggunaan param.clear() juga menunjukkan kesiapan kode untuk menerima parameter dinamis jika diinginkan nantinya. Secara keseluruhan, implementasi ini mendemonstrasikan alur kerja pelaporan berbasis JasperReports yang baik dan bisa langsung digunakan atau dikembangkan lebih lanjut sesuai kebutuhan.

BAB IV

TUGAS

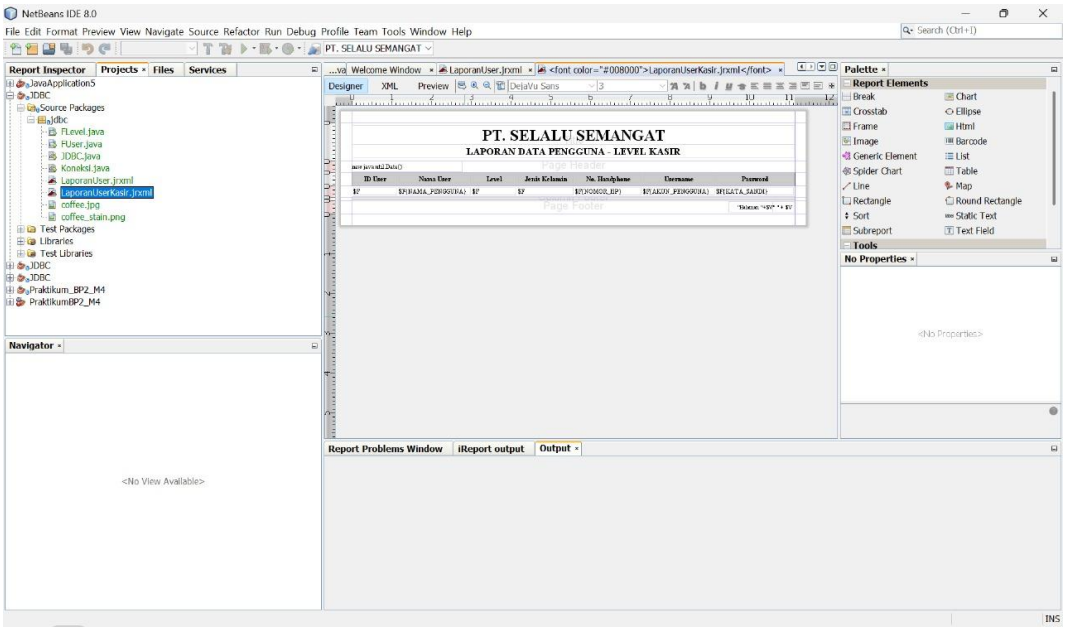
1. Buatlah laporan data user yang levelnya “Kasir”

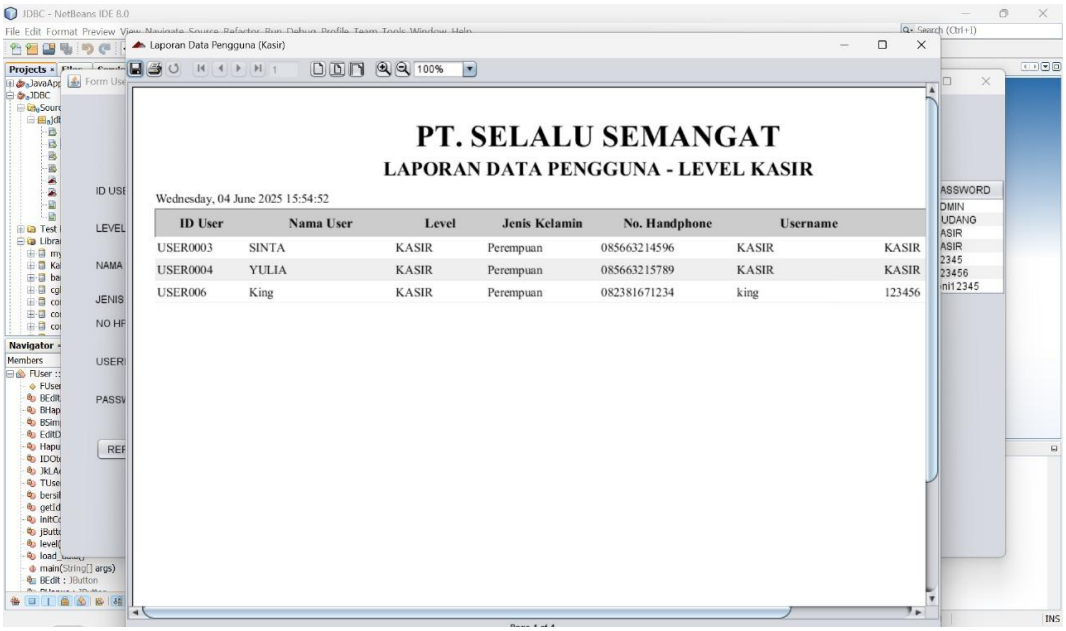
```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

    // 1. Cek Koneksi

    if (koneksi == null || koneksi.con == null) {

        JOptionPane.showMessageDialog(this, "Objek Koneksi belum
diinisialisasi atau koneksi ke database gagal! Tidak dapat mencetak report.",
"Error Koneksi", JOptionPane.ERROR_MESSAGE);

        System.err.println("jButton4ActionPerformed: Koneksi.con is null.");

        return;

    }

    try {

        // Cek status koneksi sekali lagi untuk memastikan

        if (koneksi.con.isClosed()) {

            JOptionPane.showMessageDialog(this, "Koneksi ke database
tertutup! Tidak dapat mencetak report.", "Error Koneksi",
JOptionPane.ERROR_MESSAGE);

            System.err.println("jButton4ActionPerformed: Koneksi.con is
closed.");

            return;

        }

        System.out.println("jButton4ActionPerformed: Koneksi.con status
valid.");


    } catch (SQLException e) {

        JOptionPane.showMessageDialog(this, "Error saat memeriksa status
koneksi: " + e.getMessage(), "Error Koneksi",
JOptionPane.ERROR_MESSAGE);

        e.printStackTrace();

        return;

    }
```

```
// 2. Memuat File Laporan (.jrxml)
InputStream reportStream = null;
String namaFileLaporan = "/jdbc/LaporanUserKasir.jrxml"; // Path resource relatif terhadap classpath root

try {
    System.out.println("Mencoba memuat laporan dari: " + namaFileLaporan);
    reportStream = getClass().getResourceAsStream(namaFileLaporan);

    if (reportStream == null) {
        // Jika gagal sebagai resource, coba sebagai file (kurang direkomendasikan untuk deployment)
        System.err.println("Gagal memuat laporan sebagai resource stream. Mencoba sebagai file...");
        File fileReport = new File("src/jdbc/LaporanUserKasir.jrxml"); // Path relatif terhadap root proyek
        if (!fileReport.exists()) {
            JOptionPane.showMessageDialog(this, "File laporan LaporanUserKasir.jrxml tidak ditemukan di:\n" + fileReport.getAbsolutePath() + "\nAtaupun sebagai resource: " + namaFileLaporan, "Error File Laporan", JOptionPane.ERROR_MESSAGE);
            System.err.println("File laporan tidak ditemukan di path file: " + fileReport.getAbsolutePath() + " maupun sebagai resource.");
            return;
        }
        System.out.println("Berhasil menemukan file laporan di: " + fileReport.getAbsolutePath());
        JasDes = JRXmlLoader.load(fileReport.getAbsolutePath());
    } else {
        System.out.println("Berhasil memuat laporan sebagai resource stream.");
```

```
            JasDes = JRXmlLoader.load(reportStream);
        }

        // 3. Mengkompilasi dan Mengisi Laporan
        System.out.println("Mengompilasi laporan...");
        param.clear(); // Bersihkan parameter
        JasRep = JasperCompileManager.compileReport(JasDes);
        System.out.println("Laporan berhasil dikompilasi. Mengisi
laporan...");
        JasPri = JasperFillManager.fillReport(JasRep, param, koneksi.con);
        System.out.println("Laporan berhasil diisi.");

        // 4. Menampilkan Laporan
        if (JasPri.getPages().isEmpty()) {
            JOptionPane.showMessageDialog(this, "Laporan berhasil dibuat,
tetapi tidak ada data yang sesuai dengan kriteria (Level Kasir).", "Informasi
Laporan", JOptionPane.INFORMATION_MESSAGE);
            System.out.println("Laporan kosong (tidak ada data kasir).");
            // Tetap tampilkan viewer agar pengguna tahu proses berhasil tapi
data kosong
        }

        System.out.println("Menampilkan JasperViewer...");
        JasperViewer viewer = new JasperViewer(JasPri, false);
        viewer.setTitle("Laporan Data Pengguna (Kasir)");

viewer.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_O
N_CLOSE); // Pastikan viewer bisa ditutup tanpa menutup aplikasi utama
        viewer.setVisible(true);
        System.out.println("JasperViewer seharusnya sudah tampil.");
```

```
        } catch (JRException e) {
            JOptionPane.showMessageDialog(this, "Gagal memproses laporan JasperReports: " + e.getMessage(), "Error Laporan JRE", JOptionPane.ERROR_MESSAGE);
            e.printStackTrace();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Terjadi kesalahan umum saat membuat laporan: " + e.getMessage(), "Error Umum", JOptionPane.ERROR_MESSAGE);
            e.printStackTrace();
        } finally {
            if (reportStream != null) {
                try {
                    reportStream.close();
                    System.out.println("Resource stream laporan ditutup.");
                } catch (IOException e) {
                    System.err.println("Error menutup resource stream laporan:");
                    e.printStackTrace();
                }
            }
        }
    }
```





Analisis : Kode dan tampilan laporan pada gambar menunjukkan implementasi yang lebih spesifik dari sistem pelaporan menggunakan JasperReports, yaitu menampilkan data pengguna dengan level tertentu dalam hal ini "KASIR". Dari sisi kode, proses diawali dengan validasi koneksi database yang sangat matang: tidak hanya memeriksa objek koneksi null, tetapi juga memastikan koneksi tidak tertutup (isClosed()), yang merupakan praktik sangat baik untuk menghindari error runtime yang sulit dilacak. File laporan .jrxml dicoba diakses terlebih dahulu sebagai resource melalui getClass().getResourceAsStream(), dan jika gagal, dilakukan fallback ke sistem file lokal menggunakan File, yang memperlihatkan fleksibilitas tinggi dalam pengelolaan jalur file, sangat berguna dalam perbedaan antara lingkungan pengembangan dan produksi.

Hasil akhir laporan yang ditampilkan menunjukkan hanya pengguna dengan level "KASIR" seperti SINTA, YULIA, dan King yang masuk dalam daftar. Ini membuktikan bahwa laporan telah difilter dengan benar, meskipun filtering tidak tampak eksplisit di kode Java—besar kemungkinan dilakukan langsung dalam query data pada JRXML. Judul laporan pun telah disesuaikan menjadi "LAPORAN DATA PENGGUNA LEVEL KASIR" untuk membedakannya dari laporan umum. Tidak hanya fungsional, namun juga secara user experience sudah sangat baik karena adanya pengecekan jika halaman laporan kosong dan memberikan feedback kepada pengguna melalui dialog informasi. Dengan pendekatan modular dan robust ini, kode dapat dengan mudah dikembangkan untuk laporan-laporan berdasarkan kriteria lainnya.

BAB V

KESIMPULAN

JasperReports merupakan salah satu pustaka pelaporan open-source paling populer yang digunakan dalam pengembangan aplikasi Java, khususnya untuk menghasilkan laporan yang profesional, dinamis, dan dapat dicetak. Dalam implementasinya, JasperReports sangat cocok digunakan dalam aplikasi desktop berbasis Java seperti sistem kasir, karena dapat diintegrasikan langsung dengan database menggunakan JDBC dan menghasilkan laporan dalam berbagai format (PDF, HTML, Excel, dsb.).

Dari analisis kode dan tampilan laporan yang telah dibahas, JasperReports terbukti mampu menyajikan data secara rapi, terstruktur, dan mudah dipahami oleh pengguna. Kode yang digunakan memperlihatkan tahapan penting dalam proses pelaporan, yaitu: pengecekan koneksi database, pemuatan file laporan (.jrxml) baik dari classpath maupun dari filesystem, proses kompilasi desain laporan, pengisian data laporan dengan parameter dan koneksi, hingga akhirnya menampilkan laporan menggunakan JasperViewer. Penanganan error juga dilakukan dengan baik untuk mengantisipasi berbagai kemungkinan, seperti file tidak ditemukan, koneksi database terputus, atau laporan kosong.

Laporan-laporan yang dihasilkan baik yang menampilkan semua pengguna maupun yang difilter berdasarkan kriteria tertentu seperti level pengguna (misalnya: "KASIR") menunjukkan bahwa JasperReports mampu mengakomodasi kebutuhan pelaporan berbasis parameter. Desain laporan pun dapat disesuaikan menggunakan iReport Designer atau Jaspersoft Studio, yang mendukung tampilan visual dan fleksibel dalam mendesain layout laporan.

Dengan segala kemampuannya, JasperReports tidak hanya mendukung keperluan pencetakan laporan tetapi juga meningkatkan nilai profesionalitas aplikasi secara keseluruhan. Integrasi yang erat dengan Java serta fleksibilitas dalam pemrosesan data membuatnya menjadi pilihan ideal untuk pengembangan aplikasi skala kecil hingga menengah yang membutuhkan fitur pelaporan yang andal dan mudah dikustomisasi.

Link Github Untuk Kode Lengkapnya :
https://github.com/MuhammadRizalNurfirdaus/BP2.git