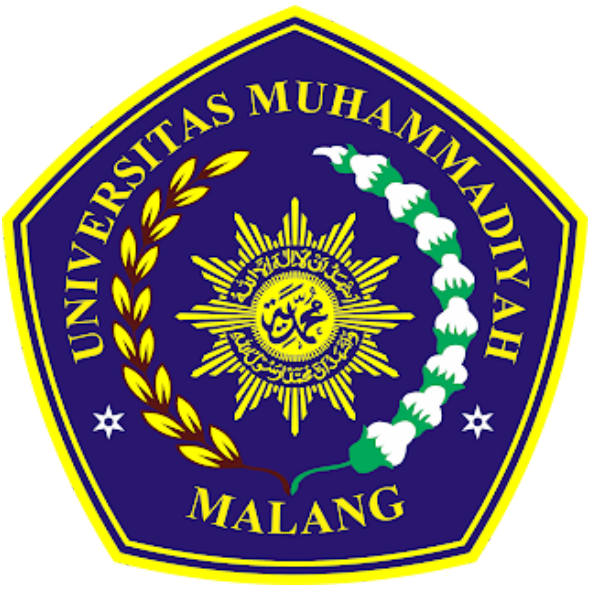
LAPORAN UJIAN AKHIR PRAKTIKUM PEMROGRAMAN LANJUT

Program *Sistem Donasi*

Laporan disusun untuk memenuhi tugas Ujian Akhir Praktikum Pemrograman Lanjut



Nama Anggota:

Syahrial Nur Faturrahman	202410370110009
Farid Al Farizi	202410370110017

Kelas: Pemrograman Lanjut B

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH MALANG

DAFTAR ISI

1. PENDAHULUAN

Aplikasi Sistem Donasi merupakan aplikasi dekstop berbasis Java yang dikembangkan sebagai bagian dari Ujian Akhir Praktikum mata kuliah Pemrograman Lanjut. Aplikasi ini dirancang untuk membantu pengelolaan data donasi secara terstruktur, mulai dari pendataan donor, pencatatan donasi masuk, hingga pengelolaan penyaluran donasi.

Pengembangan aplikasi ini bertujuan untuk mengimplementasikan konsep pemrograman penggunaan GUI Java Swing, serta CRUD (Create, Read, Update, Delete) dengan media penyimpanan file CSV.

2. TUJUAN

Tujuan dari pengembangan aplikasi Sistem Donasi ini sebagai berikut:

1. Mengimplementasikan konsep pemrograman Java berbasis GUI.
2. Menerapkan konsep CRUD dalam pengelolaan data.
3. Menggunakan file CSV sebagai media penyimpanan data.
4. Mengembangkan aplikasi dekstop menggunakan Java Swing.
5. Melakukan pengujian manual terhadap seluruh fitur aplikasi.

3. SPESIFIKASI TEKNIS PROGRAM

3.1. Graphical User Interface (GUI)

Aplikasi Sistem Donasi dikembangkan menggunakan Java Swing sebagai framework Graphical User Interface (GUI). Seluruh tampilan aplikasi dibangun menggunakan komponen Java Swing seperti JFrame, JPanel, JButton, JTable, JTextField, dan komponen pendukung lainnya.

Desain antarmuka aplikasi dibuat dengan memperhatikan aspek **user-friendly**, kemudahan navigasi, dan keterbacaan informasi. Aplikasi memiliki beberapa halaman utama yang saling terhubung melalui menu navigasi, yaitu:

1. Halaman Dashboard, yang berfungsi sebagai halaman utama dan pusat navigasi aplikasi.
2. Halaman List Data, yang menampilkan data dalam bentuk tabel dengan fitur pencarian dan pengurutan.
3. Halaman List Data, yang menampilkan data dalam bentuk tabel dengan fitur pencarian dan pengurutan.

4. Halaman Laporan/History, yang menampilkan ringkasan dan riwayat transaksi donasi dan distribusi.

3.2. Fungsionalitas Data (CRUD dan File Handling)

Aplikasi Sistem Donasi telah mengimplementasikan operasi CRUD (Create, Read, Update, Delete) pada pengelolaan data donor, donasi, dan distribusi donasi.

1. Operasi Create digunakan untuk menambahkan data baru melalui form input.
2. Operasi Read digunakan untuk menampilkan data dalam tabel.
3. Operasi Update digunakan untuk mengubah data yang telah tersimpan.
4. Operasi Update digunakan untuk mengubah data yang telah tersimpan.

Untuk menjaga persistensi data, aplikasi menggunakan file handling dengan format file CSV. Data yang dimasukkan pengguna akan disimpan ke dalam file CSV dan dimuat kembali saat aplikasi dijalankan ulang, sehingga data tidak hilang meskipun aplikasi ditutup.

3.3. Kualitas Kode dan Exception Handling

Dalam pengembangan aplikasi, telah diterapkan exception handling menggunakan mekanisme try-catch. Exception handling digunakan untuk menangani kesalahan seperti input data yang tidak valid, kesalahan pembacaan file CSV, serta kondisi ketika file penyimpanan tidak ditemukan.

Penerapan exception handling bertujuan untuk menjaga kestabilan aplikasi dan mencegah terjadinya error yang dapat menyebabkan aplikasi berhenti secara tiba-tiba.

4. IMPLEMENTASI APLIKASI

Aplikasi Sistem Donasi diimplementasikan dengan membagi struktur program ke dalam beberapa package sesuai dengan tanggung jawab masing-masing komponen. Pembagian ini bertujuan untuk menjaga keteraturan kode dan memudahkan proses pengembangan serta pemeliharaan aplikasi.

Struktur aplikasi terdiri dari package App sebagai entry point aplikasi, Model untuk merepresentasikan entitas data, Repo untuk pengelolaan data dan file CSV, UI untuk tampilan antarmuka pengguna, serta Util sebagai kumpulan class pendukung.

5. TESTING

A. Dahsboar Navigasi

Fitur	Skenario Pengujian	Langkah Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
Dashboard	Aplikasi dijalankan	Jalankan aplikasi melalui App.java	Dashboard tampil tanpa error	Sesuai	Berhasil
Navigasi	Akses menu Donor	Klik menu List Donatur pada sidebar	Halaman List Donatur tampil	Sesuai	Berhasil
Navigasi	Akses menu Donasi	Klik menu Donasi Masuk	Halaman Donasi Masuk tampil	Sesuai	Berhasil
Navigasi	Akses Menu Penyaluran	Klik menu Penyaluran	Halaman Penyaluran tampil	Sesuai	Berhasil
Navigasi	Akses menu Laporan	Klik menu Laporan/History	Halaman laporan tampil	Sesuai	Berhasil

B. Manajemen Donor

Fitur	Skenario Pengujian	Langkah Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
Donor	Tambah donor	Isi form donor lalu simpan	Data tersimpan dan muncul di tabel	Sesuai	Berhasil
Donor	Edit donor	Pilih donor lalu edit	Data donor berubah	Sesuai	Berhasil
Donor	Hapus donor	Pilih donor lalu hapus	Data donor terhapus	Sesuai	Berhasil
Donor	Pencarian donor	Cari donor berdasarkan nama	Data terfilter	Sesuai	Berhasil
Donor	Sorting data	Klik menu sort	Data terurut	Sesuai	Berhasil

C. Manajemen Donasi

Fitur	Skenario Pengujian	Langkah Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
Donasi	Tambah donasi	Isi form donasi lalu simpan	Data donasi tersimpan	Sesuai	Berhasil
Donasi	Edit donasi	Memilih data lalu edit	Data donasi berubah	Sesuai	Berhasil
Donasi	Hapus donasi	Memilih data lalu hapus	Data donasi terhapus	Sesuai	Berhasil
Donasi	Validasi nominal	Mengisi huruf pada nominal	Input ditolak	Sesaui	Berhasil
Donasi	Validasi data kosong	Menyimpan form kosong	Proses ditolak	Sesuai	Berhasil
Donasi	Sorting Data	Klik menu sort	Data terurut	Sesuai	Berhasil
Donasi	Pencarian	Cari berdasarkan ID	Daftar terfilter	Sesuai	Berhasil

D. Manajemen Penyaluran Donasi

Fitur	Skenario Pengujian	Langkah Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
Penyaluran	Tambah Penyaluran	Mengisi form lalu simpan	Data penyaluran tersimpan	Sesuai	Berhasil
Penyaluran	Edit Penyaluran	Memilih data lalu edit	Data berubah	Sesuai	Berhasil
Penyaluran	Hapus Penyaluran	Memilih data lalu hapus	Data terhapus	Sesuai	Berhasil
Penyaluran	Input keterangan	Mengisi keterangan penyaluran	Data tersimpan	Sesuai	Berhasil
Penyaluran	Sorting Data	Klik menu sort	Data terurut	Sesuai	Berhasil
Penyaluran	Pencarian	Cari berdasarkan ID	Daftar terfilter	Sesuai	Berhasil

E. Pengujian File Handling (CSV)

Fitur	Skenario Pengujian	Langkah Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
File CSV	Simpan data	Menambah data lalu menutup aplikasi	Data tersimpan di file CSV	Sesuai	Berhasil
File CSV	Load data	Membuka ulang aplikasi	Data tetap ada	Sesuai	Berhasil

6. CODE REVIEW

a. DonorRepository.java

1. Kode asli:

```
catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Gagal membaca donors.csv:\n" + e.getMessage(),
            "Error", JOptionPane.ERROR_MESSAGE);
}
```

Masalah:

Layer Repository langsung berinteraksi dengan UI menggunakan JOptionPane. Hal ini melanggar prinsip *Separation of Concerns*, karena Repository seharusnya hanya menangani data, bukan tampilan.

Solusi:

Repository sebaiknya melempar exception ke layer pemanggil (UI), sehingga penanganan pesan error dilakukan di UI.

```
catch (Exception e) {
    throw new RuntimeException("Gagal membaca donors.csv", e);
}
```

2. Kode asli:

```
 donors.add(new Donor(
        p[0], p[1], p[2], p[3], LocalDate.parse(p[4])
));
```

Masalah:

Tidak terdapat validasi eksplisit terhadap format tanggal atau data kosong sebelum membuat objek Donor.

Solusi:

Tambahkan validasi sederhana sebelum parsing data.

```
  if (!p[4].isEmpty()) {
    donors.add(new Donor(
            p[0], p[1], p[2], p[3], LocalDate.parse(p[4])
    ));
}
```

b. DistributionRepository.java

1. Kode asli:

```
String[] p = line.split(",", -1);
```

Masalah:

Parsing CSV menggunakan split(",") rentan terhadap kesalahan apabila data mengandung tanda koma, misalnya pada kolom catatan atau penerima.

Dampak: Data dapat terbaca tidak sesuai kolom dan menyebabkan kesalahan parsing.

Solusi:

Gunakan utilitas parsing CSV yang lebih aman atau memusatkan parsing pada class CsvUtil.

```
String[] p = CsvUtil.parseLine(line);
```

2. Kode asli:

```
LocalDate tgl = LocalDate.parse(p[1]);
```

Masalah:

Validasi format tanggal tidak dilindungi, Jika format tanggal pada CSV tidak sesuai standar ISO (yyyy-MM-dd), aplikasi akan gagal mem-parsing tanggal. Ini spesifik Distribution, karena donor biasanya tanggal dibuat otomatis distribution bisa diinput manual / diedit.

Solusi:

Gunakan parsing aman dengan `try-catch` atau DateTimeFormatter.

```
LocalDate tgl;
try {
    tgl = LocalDate.parse(p[1]);
} catch (Exception e) {
    tgl = LocalDate.now();
}
```

c. DonationRepository.java

1. Kode asli:

```
 nominal = Double.parseDouble(p[5]);

jumlahBarang = Integer.parseInt(p[7]);
```

Masalah:

Repository tidak memvalidasi apakah `nominal` bernilai negatif `jumlahBarang` bernilai negatif. Akibatnya, data tidak logis masih bisa diproses jika berasal dari file CSV.

Solusi:

Tambahkan validasi nilai minimum.

```
 if (nominal < 0) nominal = 0;
if (jumlahBarang < 0) jumlahBarang = 0;
```

2. Kode asli:

```
bw.write("donasiId,tanggal,donorId,jenis,kategori,nominal,namaBarang,jumlahBarang,catatan\n");
```

Masalah:

Header CSV ditulis secara hardcode, sehingga sulit dirawat, rentan inkonsistensi jika format berubah.

Solusi:

Gunakan konstanta untuk header CSV.

```
private static final String CSV_HEADER =
    "donasiId,tanggal,donorId,jenis,kategori,nominal,namaBarang,jumlahBarang,catatan";
```

d. Distribution.java

Kode asli:

```
private String jenis; // UANG / BARANG
```

Masalah:

Atribut jenis menggunakan tipe String, sehingga memungkinkan nilai yang tidak konsisten (misalnya "uang", "Uang", atau nilai lain yang tidak valid). Hal ini berpotensi menyebabkan inkonsistensi data, kesalahan logika pada proses validasi di layer lain

Solusi:

Gunakan enum untuk membatasi nilai jenis distribusi agar lebih aman dan konsisten.

```
public enum JenisDistribusi {
    UANG, BARANG
}
```

dan atribut diubah menjadi:

```
private JenisDistribusi jenis;
```

e. Donation.java

Kode asli:

```
private String donorId;
```

Masalah:

Class Donation hanya menyimpan donorId sebagai String tanpa validasi atau keterkaitan langsung dengan entitas Donor. Hal ini menyebabkan kemungkinan donorId kosong atau tidak valid, relasi data tidak konsisten. Ini khas Donation, karena distribution tidak berelasi ke Donor, Donor tidak bergantung ke entitas lain.

Solusi:

Tambahkan validasi sederhana atau mekanisme pengecekan relasi di level model atau service.

```
public void setDonorId(String donorId) {
    if (donorId != null && !donorId.isEmpty()) {
        this.donorId = donorId;
    }
}
```