

TUGAS BESAR
“SplitIt: Smart Expense Splitter”



Disusun Oleh:

Edelweiss Salsabilla - 103052300038
Muthia Rezi Aisyah - 103052300114
Amalia Ananda Putri - 103052330078
Alishadena C.N.R.H - 103052300032
Annisa Azzahra Rahmah - 103052300056

PEMROGRAMAN BERORIENTASI OBJEK
PROGRAM STUDI S1 SAINS DATA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
2026

DAFTAR ISI

PENDAHULUAN.....	3
1.1 Latar Belakang.....	3
1.2 Rumusan Masalah.....	3
1.3 Tujuan Pengembangan.....	3
1.4 Manfaat Aplikasi.....	4
1.5 Ruang Lingkup.....	4
RANCANGAN ARSITEKTUR DAN KONSEP PBO.....	5
2.1. Arsitektur Aplikasi (Strategy Pattern).....	5
2.2. Penerapan Konsep PBO.....	5
2.2.1. Inheritance dan Abstract Class.....	5
2.2.2. Interface dan Polymorphism.....	5
2.3. Prinsip PBO yang Digunakan.....	5
RANCANGAN FITUR DAN IMPLEMENTASI.....	7
3.1. Daftar Fitur (List Features).....	7
3.2. Diagram Kelas (Class Diagram).....	8
3.3. Deskripsi Detail Kelas.....	9
IMPLEMENTASI FITUR DAN PEMBAGIAN TUGAS.....	10
HASIL APLIKASI.....	13
5.1. Halaman Dashboard.....	13
5.2. Halaman Buat/Lihat Group.....	13
5.3. Halaman Detail Group + Anggota.....	14
5.4. Halaman Tambah Transaksi Even (Bagi Rata).....	15
5.5. Halaman Tambah Transaksi Uneven (Bagi Tidak Rata/Per Item + Biaya Tambahan).....	16
5.6. Halaman Summary.....	18
5.7. Demo Website.....	21
PENUTUP.....	22
6.1. Kesimpulan.....	22

PENDAHULUAN

1.1 Latar Belakang

Dalam lingkungan akademik dan sosial, mahasiswa seringkali terlibat dalam kegiatan kelompok yang memerlukan pengelolaan biaya bersama (patungan). Kegiatan seperti makan bersama, membeli perlengkapan tugas, atau berlibur bersama seringkali menimbulkan kebingungan saat proses pembagian tagihan. Perhitungan manual rentan terhadap kesalahan (*human error*), tidak transparan, dan memakan waktu yang cukup lama, terutama jika metode pembagiannya kompleks (misalnya, ada yang membayar lebih dulu, ada yang tidak ikut dalam satu item atau kondisi, dan lainnya).

Kondisi tersebut dapat menimbulkan kecanggungan sosial dan potensi konflik kecil. Oleh karena itu, untuk mengatasi masalah ini, diusulkan pengembangan aplikasi berbasis Pemrograman Berorientasi Objek (PBO) yang diberi nama “SplitIt: *Smart Expense Splitter*”.

Aplikasi SplitIt dirancang untuk menyederhanakan dan mengotomatisasi seluruh proses *split bill* (patungan). Pengguna dapat membuat grup, mencatat setiap transaksi, dan membiarkan sistem menghitung secara adil dan transparan siapa yang berhutang kepada siapa hingga semua neraca keuangan impas (*settled*).

1.2 Rumusan Masalah

1. Bagaimana merancang sistem yang dapat mencatat transaksi pengeluaran dalam sebuah grup?
2. Bagaimana mengimplementasikan logika pembagian biaya yang fleksibel, mencakup pembagian rata (*even*) dan tidak rata (*uneven*)?
3. Bagaimana menerapkan konsep PBO (*inheritance*, *abstract class*, *interface*, dan lainnya) secara efektif untuk membangun arsitektur yang kokoh?
4. Bagaimana sistem dapat mengkalkulasi dan menyajikan rangkuman hutang-piutang yang sederhana dan mudah dipahami?

1.3 Tujuan Pengembangan

Tujuan dari pengembangan aplikasi SplitIt antara lain:

1. Membangun aplikasi yang mampu mengelola grup, anggota, dan mencatat transaksi keuangan.
2. Mengimplementasikan *design pattern* (*strategy pattern*) menggunakan *interface* untuk menangani berbagai metode pembagian biaya.
3. Menerapkan *inheritance* dan *abstract class* untuk menciptakan model data yang konsisten.
4. Menghasilkan kalkulator hutang yang dapat menyederhanakan alur pembayaran antar anggota.

1.4 Manfaat Aplikasi

1. Bagi Pengguna
Memberikan kemudahan, transparansi, dan keadilan dalam mengelola keuangan kelompok, serta mengurangi potensi konflik sosial.
2. Bagi Pengembang
Mengaplikasikan secara praktis teori dan konsep Pemrograman Berorientasi Objek (PBO) dalam sebuah proyek nyata.

1.5 Ruang Lingkup

Untuk memastikan proyek ini layak (*feasible*) dan dapat diselesaikan sesuai *timeline* mata kuliah, maka ditetapkan batasan (ruang lingkup) sebagai berikut.

1. Fokus Utama
Logika bisnis (OOP) di *backend*.
2. Yang akan Dibuat
Manajemen grup, manajemen anggota, pencatatan transaksi (dengan tipe bagi rata (*even*) dan bagi tidak rata (*uneven*)), kalkulasi rangkuman hutang, serta koneksi ke *database* (phpMyAdmin).
3. Yang Tidak akan Dibuat
Sistem *login* atau *register* (data pengguna bersifat lokal), fitur *payment gateway*, dan sinkronisasi antar perangkat.

RANCANGAN ARSITEKTUR DAN KONSEP PBO

2.1. Arsitektur Aplikasi (*Strategy Pattern*)

Aplikasi ini akan dirancang menggunakan *Strategy Design Pattern*. *Pattern* ini dipilih untuk menangani inti permasalahan, yaitu metode pembagian biaya yang berbeda.

Pada aplikasi ini, akan dipisahkan "konteks" (*transaction*) dari "strategi" (*ISplitStrategy*). *Transaction* akan memiliki sebuah referensi ke *interface* *ISplitStrategy*. Saat kalkulasi, *transaction* akan mendelegasikan pekerjaan ke strategi mana pun yang sedang digunakannya (*EvenSplitStrategy* atau *UnevenSplitStrategy*), tanpa harus tahu detail implementasinya.

2.2. Penerapan Konsep PBO

Proyek ini memenuhi syarat dari Pemrograman Berorientasi Objek (PBO) sebagai berikut.

2.2.1. *Inheritance* dan *Abstract Class*

- a. Penerapan
Melalui sebuah *abstract class* bernama *BaseEntity*.
- b. Tujuan
Kelas ini akan berfungsi sebagai *parent class* untuk semua model data utama (*User*, *Group*, dan *Transaction*).
- c. Detail
BaseEntity akan memiliki atribut *protected String id* dan *method* konkret *public String getId()*. Ini memastikan bahwa setiap entitas data utama dalam sistem pasti memiliki ID yang unik sehingga memenuhi prinsip *inheritance* (pewarisan atribut dan *method*).

2.2.2. *Interface* dan *Polymorphism*

- a. Penerapan melalui sebuah *interface* bernama *ISplitStrategy*.
- b. *Interface* ini akan mendefinisikan "kontrak" atau *blueprint* untuk semua algoritma pembagian biaya.
- c. Detail *ISplitStrategy* akan memiliki satu *method* abstrak, yaitu *public Map<User, Double> calculateShares(...)*.
- d. *Polymorphism* kelas *Transaction* akan memiliki atribut *private ISplitStrategy strategy*. Saat *Transaction.executeSplit()* dipanggil, ia akan memanggil *strategy.calculateShares()*. Perilaku yang terjadi akan berbeda-beda (polimorfik) tergantung pada objek konkret (*EvenSplitStrategy* atau *UnevenSplitStrategy*) yang di-*inject* ke dalam *Transaction* tersebut.

2.3. Prinsip PBO yang Digunakan

Prinsip Pemrograman Berorientasi Objek (PBO) yang akan digunakan dalam aplikasi ini adalah sebagai berikut.

1. *Encapsulation*

Atribut dibuat *private/protected* dan diakses lewat *getter* dan/atau *setter*.

2. *Abstraction*

Kelas *BaseEntity* dan *ISplitStrategy* memisahkan konsep dari implementasi.

3. *Inheritance*

User, *Group*, *Transaction* mewarisi *BaseEntity*.

4. *Polymorphism*

ISplitStrategy memungkinkan strategi pembagian bervariasi tanpa ubah kode utama.

5. *Composition*

Group memiliki daftar *User* dan *Transaction* yang akan dihapus bersamaan.

6. *Collection*

Penggunaan *List* dan *Map*, yaitu struktur data dinamis untuk menyimpan banyak objek.

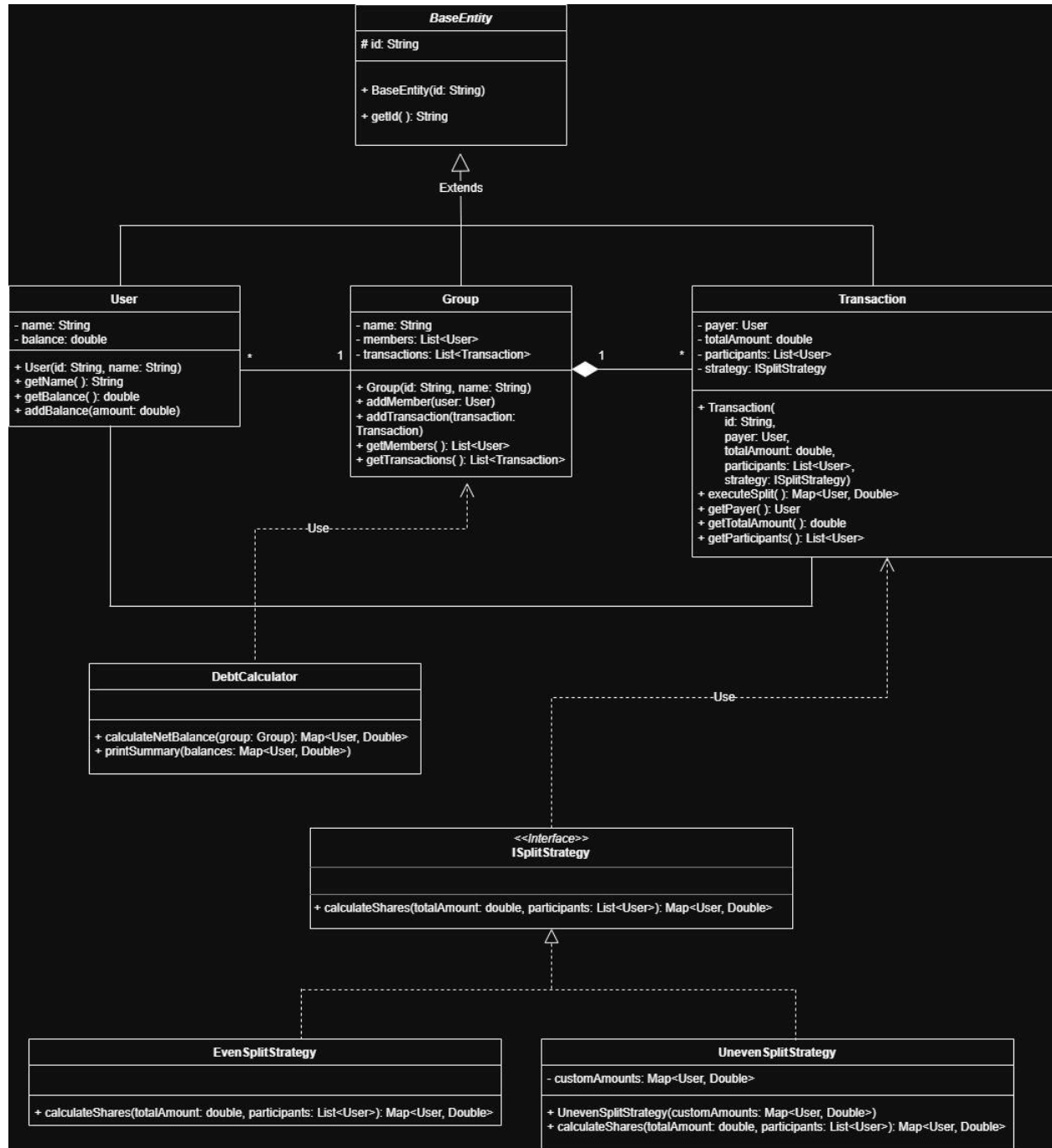
List digunakan untuk menyimpan data berurutan, sedangkan *Map* menyimpan pasangan *key-value* (contohnya *User* sebagai *key* dan jumlah uang sebagai *value*).

RANCANGAN FITUR DAN IMPLEMENTASI

3.1. Daftar Fitur (*List Features*)

1. F1: Manajemen Grup
Membuat grup baru dan menambahkan anggota/pengguna (*User*) ke dalamnya.
2. F2: Pencatatan Transaksi
Mencatat pengeluaran baru, mencakup siapa yang membayar, berapa totalnya, dan siapa saja yang terlibat.
3. F3: Strategi Bagi Rata (*Even Split*)
Logika untuk membagi totalAmount secara merata ke semua partisipan yang dipilih.
4. F4: Strategi Bagi Tidak Rata (*Uneven Split*)
Logika untuk membagi totalAmount berdasarkan nominal kustom yang diinput manual oleh pengguna untuk setiap partisipan.
5. F5: Kalkulator Hutang dan Rangkuman
Mesin yang menghitung neraca keuangan semua anggota dalam satu grup dan menyederhanakannya menjadi daftar "siapa-bayar-siapa".

3.2. Diagram Kelas (Class Diagram)



3.3. Deskripsi Detail Kelas

Kelas	Deskripsi	Relasi / Konsep
BaseEntity	Kelas abstrak berisi id dan getId() untuk konsistensi identitas tiap entitas.	<i>Abstract (Superclass)</i>
User	Mewakili anggota dalam grup. Menyimpan nama dan saldo hasil perhitungan <i>split</i> .	<i>Inheritance</i> dari BaseEntity
Group	Kumpulan anggota dan transaksi. Mengatur logika menambah anggota dan menghitung ringkasan hutang.	<i>Composition</i> (memiliki User dan Transaction)
Transaction	Mewakili satu pengeluaran. Menyimpan siapa yang membayar, total, dan siapa yang ikut membayar.	<i>Aggregation</i> dengan User
ISplitStrategy	<i>Interface</i> untuk mendefinisikan kontrak pembagian biaya.	<i>Interface</i>
EvenSplitStrategy	Implementasi pembagian rata.	Implementasi ISplitStrategy
UnevenSplitStrategy	Implementasi pembagian tidak rata (<i>custom</i>).	Implementasi ISplitStrategy
DebtCalculator	<i>Utility class</i> untuk menghitung dan menyederhanakan hutang antar anggota grup.	<i>Aggregation</i> ke Group

IMPLEMENTASI FITUR DAN PEMBAGIAN TUGAS

1. Edelweiss Salsabila

Peran: Desain arsitektur sistem serta implementasi BaseEntity, User, dan manajemen saldo.

Tanggung Jawab:

- Merancang struktur kelas utama pada aplikasi SplitBill
- Membuat *class* BaseEntity sebagai *parent class*
- Mengimplementasikan *class* User
- Mengelola fitur saldo (*balance*) pengguna

Hasil yang Sudah Dikerjakan:

- *Class* BaseEntity berhasil dibuat untuk menyimpan atribut umum seperti id dan createdAt
- *Class* User berhasil diimplementasikan dengan atribut:
 - *Name*
 - *Balance*
- *Method* untuk:
 - Menambah saldo
 - Mengurangi saldo
 - Melihat saldo *user*
- *Class* User telah digunakan oleh *class* lain (Transaction dan Group)

2. Muthia Rezi Aisyah

Peran: Implementasi Transaction serta penerapan *Strategy Pattern* menggunakan ISplitStrategy.

Tanggung Jawab:

- Membuat *class* Transaction
- Menerapkan *Strategy Pattern* untuk metode pembagian tagihan
- Membuat *interface* ISplitStrategy

Hasil yang Sudah Dikerjakan:

- *Class* Transaction berhasil dibuat dengan atribut:
 - *Payer*
 - *Amount*
 - *Participants*
 - *splitstrategy*
- *Interface* ISplitStrategy berhasil dibuat sebagai kontrak pembagian
- Transaction dapat menggunakan *strategy* berbeda tanpa mengubah *logic* utama.

3. Amalia Ananda Putri

Peran: Pembuatan *EvenSplitStrategy* dan *UnevenSplitStrategy* serta pengujian logika pembagian tagihan.

Tanggung Jawab:

- Mengimplementasikan *interface* *ISplitStrategy* pada dua jenis strategi pembagian tagihan.
- Mengembangkan logika pembagian tagihan secara merata (*Even Split*) dan tidak merata (*Uneven Split*).
- Mengelola pembagian berbasis jumlah peserta dan nominal khusus per *user*.
- Menangani validasi data input untuk mencegah kesalahan saat proses pembagian.
- Mengintegrasikan strategi pembagian ke dalam proses transaksi.

Hasil yang Sudah Dikerjakan:

- *Class EvenSplitStrategy* berhasil dibuat dengan:
 - Logika pembagian tagihan secara merata
 - Perhitungan berdasarkan jumlah peserta
- *EvenSplitStrategy* dapat:
 - Membagi total tagihan sama rata ke seluruh anggota
 - Menghasilkan nilai pembagian untuk setiap *user*
- *Class UnevenSplitStrategy* berhasil dibuat dengan:
 - Logika pembagian tagihan tidak merata
 - Penggunaan nominal khusus per *user*
- *UnevenSplitStrategy* dapat:
 - Membagi tagihan berdasarkan nilai yang ditentukan
 - Memberikan nilai *default* jika nominal *user* tidak tersedia
- Strategi pembagian telah:
 - Terintegrasi dengan proses transaksi
 - Diuji menggunakan beberapa skenario pembagian tagihan

4. Annisa Azzahra Rahmah

Peran: Implementasi Group dan pengelolaan anggota serta transaksi (komposisi).

Tanggung Jawab :

- Membuat *class* Group
- Mengelola relasi antara group, user, dan transaction
- Mengatur penambahan anggota dan transaksi ke dalam group

Hasil yang Sudah Dikerjakan:

- *Class Group* berhasil dibuat dengan:
 - Daftar anggota
 - Daftar transaksi
- Group dapat:
 - Menambah anggota
 - Menambahkan transaksi
 - Menampilkan daftar transaksi

5. Alishadena Chandrani Noor Riva Hutomo

Peran: Pembuatan *DebtCalculator*, *DashboardServlet*, dan *SummaryServlet*, integrasi perhitungan akhir sistem, serta dokumentasi dan pengujian.

Tanggung Jawab:

- Mengimplementasikan *class* DebtCalculator untuk menghitung saldo bersih (utang– piutang) setiap *user*.
- Mengintegrasikan hasil perhitungan saldo ke dalam tampilan aplikasi melalui SummaryServlet.
- Mengembangkan DashboardServlet sebagai halaman ringkasan awal aplikasi.
- Mengelola alur data antara model, *service*, dan *servlet*.
- Menyusun dokumentasi teknis serta melakukan pengujian fungsional sistem.

Hasil yang Sudah Dikerjakan:

- *Class* DebtCalculator berhasil dibuat dengan:
 - Logika perhitungan saldo bersih (utang–piutang)
 - Integrasi hasil pembagian transaksi dari *split strategy*
- DebtCalculator dapat:
 - Menghitung saldo akhir setiap anggota grup
 - Menentukan status utang atau piutang *user*
 - Menghasilkan ringkasan hasil perhitungan
- SummaryServlet berhasil dibuat untuk:
 - Mengambil data grup aktif
 - Menjalankan perhitungan saldo menggunakan DebtCalculator
 - Mengirim hasil perhitungan ke halaman *summary*
- DashboardServlet berhasil dibuat untuk:
 - Menampilkan informasi ringkasan awal aplikasi
 - Menampilkan jumlah grup yang tersedia

Detail *file* nya dapat dilihat disini:

https://drive.google.com/file/d/1LWaXrvxxazb3ZWfp0rBbxzf_KaEMK7Us/view?usp=sharing

HASIL APLIKASI

5.1. Halaman *Dashboard*

Dashboard pada aplikasi SplitIt berfungsi sebagai halaman utama yang memberikan gambaran umum kondisi aplikasi secara cepat kepada pengguna.

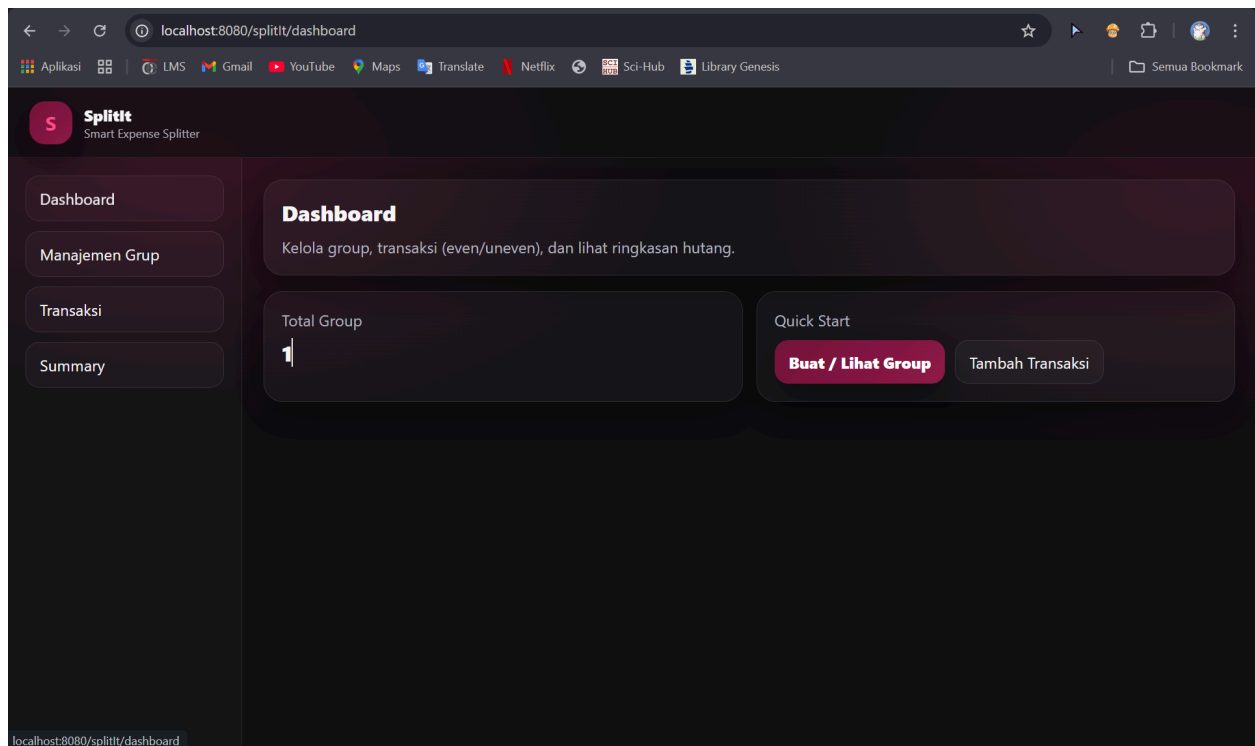
Di halaman ini, pengguna dapat:

1. Melihat ringkasan data, seperti jumlah grup yang sudah dibuat.
2. Mengetahui status awal aplikasi, apakah sudah ada grup atau belum.
3. Mengakses fitur utama dengan cepat melalui tombol *Quick Start*.

Dashboard juga menjadi pusat navigasi karena dari sini pengguna dapat langsung:

1. Masuk ke Manajemen Grup untuk membuat atau mengatur grup.
2. Menuju halaman Transaksi untuk menambahkan transaksi baru.
3. Melihat *Summary* untuk mengetahui hasil perhitungan hutang dan piutang.

Dengan adanya *dashboard*, pengguna tidak perlu langsung masuk ke halaman teknis, tetapi dapat memahami kondisi aplikasi dan menentukan langkah selanjutnya dengan lebih mudah.



5.2. Halaman *Buat/Lihat Group*

Fungsi utama halaman ini adalah sebagai pusat pengelolaan grup dalam aplikasi SplitIt. Di halaman ini, pengguna dapat membuat grup baru dan melihat daftar grup yang sudah ada.

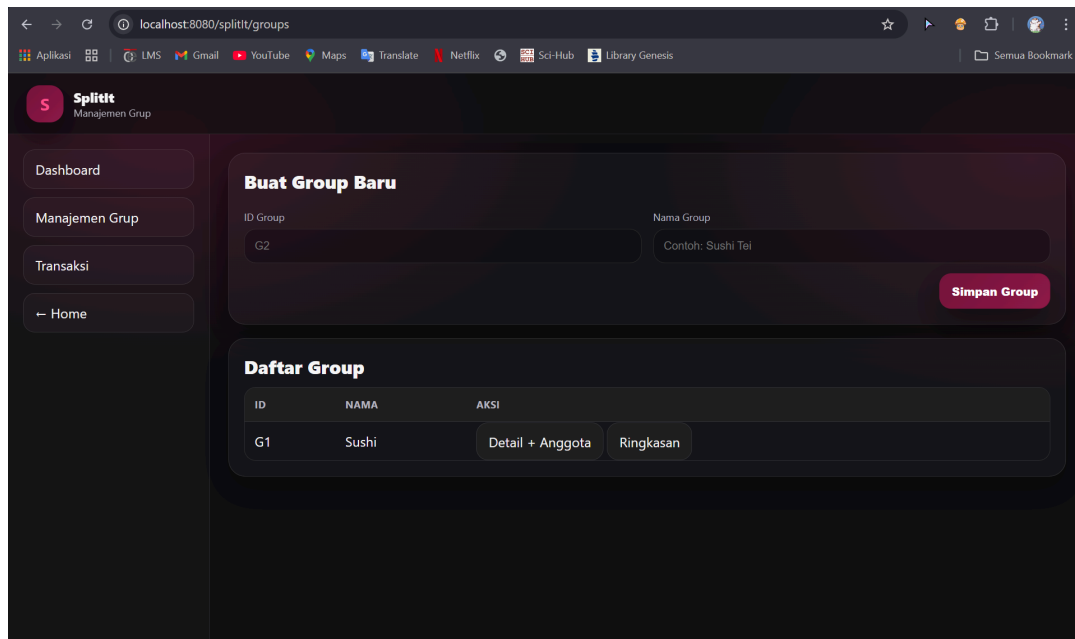
Penjelasan fungsional:

Halaman *Manajemen Grup* berfungsi untuk:

1. Membuat grup baru
 - User mengisi ID *Group* dan Nama *Group*.
 - Saat tombol Simpan *Group* ditekan, data grup disimpan ke *database*.
 - Grup ini nantinya menjadi wadah transaksi dan anggota.
2. Melihat daftar grup yang sudah dibuat
 - Menampilkan tabel berisi ID *Group* dan Nama *Group*.
 - Setiap grup punya aksi:
 - Detail + Anggota → masuk ke halaman detail grup.
 - Ringkasan → melihat ringkasan transaksi dan hutang/piutang grup tersebut.

Konsep OOP yang digunakan:

1. *Group* sebagai objek (model).
2. Servlet sebagai penghubung antara UI dan *database*.
3. DataSplit sebagai layer akses data (DAO).



5.3. Halaman Detail Group + Anggota

Fungsi utama halaman ini adalah untuk mengelola anggota (*User*) di dalam sebuah grup tertentu. Penjelasan fungsional:

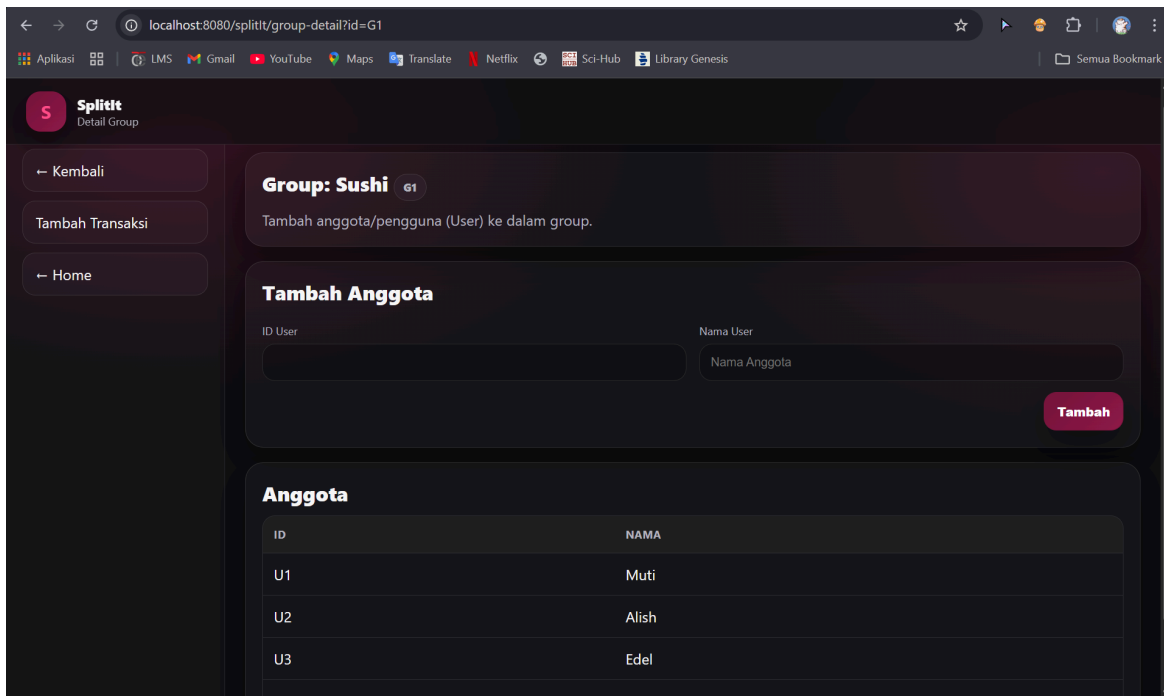
Halaman *Detail Group* digunakan untuk:

1. Menampilkan informasi grup yang dipilih
 - Nama *Group*
 - ID *Group*
 - Konteks ini penting supaya semua aksi (tambah user/transaksi) jelas masuk ke grup mana.

2. Menambahkan anggota ke grup
 - *User* mengisi:
 - ID *User*
 - Nama *User*
 - Data *user* disimpan ke tabel *users*, lalu dihubungkan ke *group* lewat *group_members*.
3. Menampilkan daftar anggota grup
 - Menampilkan semua *user* yang tergabung dalam grup.
 - Data ini dipakai lagi di halaman transaksi sebagai:
 - *Payer*
 - *Participants*

Halaman ini dibuat karena:

1. Transaksi tidak dapat dibuat tanpa grup dan anggota.
2. Semua perhitungan *split* (*Even/Uneven*) bergantung pada anggota grup ini.



5.4. Halaman Tambah Transaksi *Even* (Bagi Rata)

Halaman Tambah Transaksi (*Even Split*) digunakan untuk mencatat transaksi pengeluaran dalam suatu grup dengan metode pembagian bagi rata. Pada mode ini, total pengeluaran akan dibagi secara sama besar kepada seluruh peserta yang terlibat dalam transaksi.

User dapat menentukan:

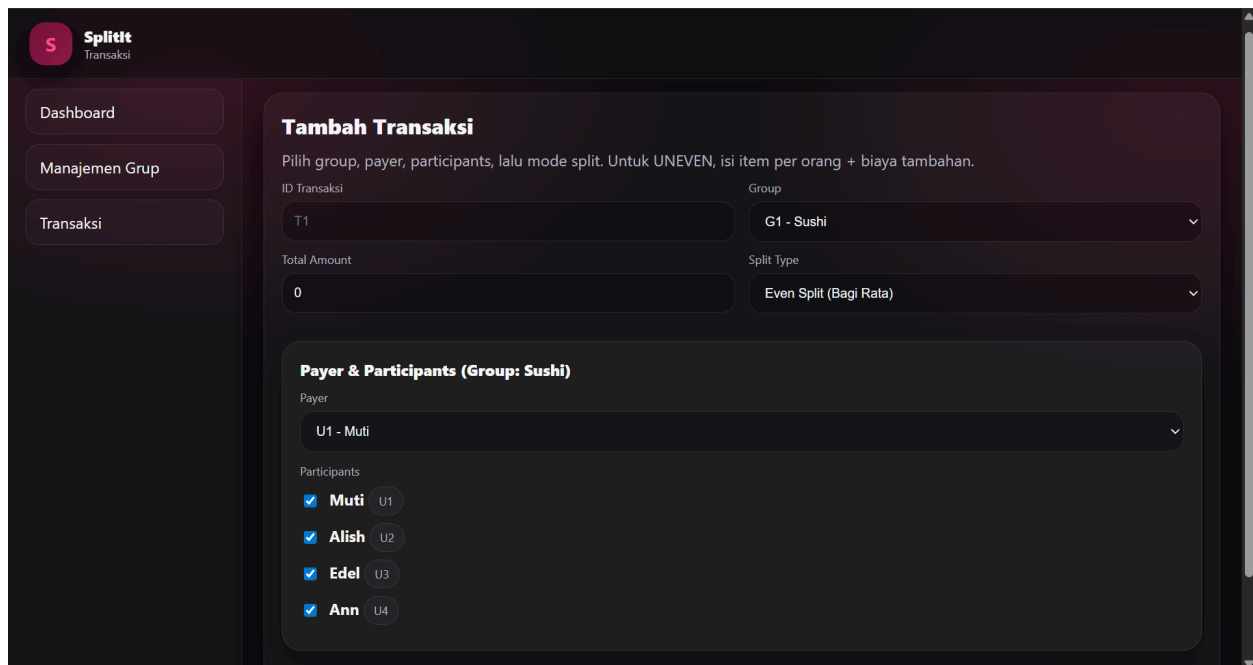
1. ID transaksi sebagai identitas unik.
2. Memilih grup tempat transaksi dicatat.
3. Memasukkan total nominal pengeluaran.

Selanjutnya, *User* memilih:

1. *Payer*, yaitu anggota yang membayar transaksi terlebih dahulu.
2. Menentukan *participants*, yaitu anggota grup yang ikut menanggung biaya tersebut.

Dalam metode *Even Split*, sistem secara otomatis menghitung jumlah yang harus dibayar oleh setiap *participant* dengan cara membagi *total amount* secara merata. Hasil perhitungan ini kemudian akan digunakan untuk menentukan posisi utang dan piutang masing-masing anggota pada halaman *summary*.

Halaman ini mempermudah pencatatan transaksi sederhana tanpa perlu memasukkan detail item per orang sehingga cocok digunakan untuk pengeluaran bersama seperti makan bersama, iuran, atau biaya kolektif lainnya.



5.5. Halaman Tambah Transaksi *Uneven* (Bagi Tidak Rata/Per Item + Biaya Tambahan)

Halaman Tambah Transaksi (*Uneven*) digunakan untuk mencatat transaksi dengan pembagian biaya tidak merata, yaitu setiap peserta dapat memiliki total pembayaran yang berbeda sesuai item yang dikonsumsi dan biaya tambahan yang dikenakan.


Pada halaman ini, *user* terlebih dahulu mengisi ID Transaksi dan memilih *Group* yang aktif. Setelah itu, *user* menentukan *payer*, yaitu orang yang membayar transaksi secara keseluruhan, serta memilih *participants*, yaitu anggota grup yang ikut terlibat dalam transaksi tersebut. Mode *Uneven Split* (Per Item + Biaya Tambahan) memungkinkan *user* memasukkan:

1. Item per orang, yaitu daftar item yang dikonsumsi oleh masing-masing peserta beserta nominal harganya.
2. Biaya tambahan, seperti pajak, *service*, atau admin, yang dapat berupa persentase maupun nominal tetap.

Sistem kemudian akan:

1. Menghitung subtotal item untuk setiap peserta berdasarkan item yang dimasukkan.
2. Menghitung total biaya tambahan dari pajak/*service*/admin.
3. Membagikan biaya tambahan ke setiap peserta sesuai aturan sistem (misalnya proporsional terhadap subtotal item).
4. Menampilkan *preview* pembagian yang berisi:
 - Subtotal item per peserta (*user*)
 - *Share* biaya tambahan per peserta (*user*)
 - Total akhir yang harus dibayar masing-masing peserta (*user*)

Preview ini bersifat *real-time* sehingga setiap perubahan pada item atau biaya tambahan langsung memperbarui hasil perhitungan. Setelah hasil pembagian dirasa sesuai, *user* dapat menyimpan transaksi dan data akan diproses oleh *backend* untuk disimpan ke *database* serta digunakan dalam perhitungan hutang dan piutang di halaman *summary*.


SplitIt
 Transaksi

Uneven Builder

Isi item per orang, lalu biaya tambahan (pajak/service/admin bisa dimasukan di sini). Sistem akan bagi biaya sesuai masukan.

Item per Orang

Bisa tambah sebanyak yang diperlukan.

Nama
 U1 - Muti

Item
 Nama Item

Nominal
 0

Hapus

Biaya Tambahan

Keterangan
 Pajak / Service / Admin

Tipe
 Percent

Nilai
 0

Hapus

Preview Pembagian (UNEVEN)

Subtotal: 0 Fee: 0 Total Akhir: 0

USER	SUBTOTAL ITEM	SHARE BIAYA	TOTAL AKHIR
U1	0	0	0
U2	0	0	0
U3	0	0	0
U4	0	0	0

Simpan Transaksi

5.6. Halaman *Summary*

Halaman *Summary* digunakan untuk menampilkan ringkasan akhir pembagian biaya dalam satu grup. Di halaman ini, sistem menggabungkan seluruh transaksi yang pernah dicatat (baik *Even Split* maupun *Uneven Split*) lalu menghitung posisi keuangan masing-masing anggota grup.

Summary membantu *user* untuk:

1. Mengetahui siapa yang punya piutang (dibayar orang lain/yang membayar secara keseluruhan di awal)
2. Mengetahui siapa yang punya utang (harus membayar)
3. Melihat detail kontribusi setiap transaksi secara transparan

Bagian-bagian di dalam halaman *Summary*:

1. *Net Balance* (Hutang/Piutang)

Bagian *Net Balance* menunjukkan akumulasi saldo bersih tiap anggota grup dari seluruh transaksi.

Cara kerja konsepnya:

- Nilai positif (Piutang) → anggota tersebut membayar lebih besar dibanding kewajibannya atau dengan kata lain anggota tersebut yang membayar semuanya di awal.
- Nilai negatif (Utang) → anggota tersebut masih harus membayar ke anggota grup lainnya.

Contoh dari tampilan:

- Muti (U1) – 619.255,00 (Piutang)
Artinya Muti sering menjadi *payer* dan total uang yang ia keluarkan lebih besar dari yang seharusnya ia tanggung sendiri.
- Alish, Edel, Ann – nilai negatif (Utang)
Artinya mereka memiliki kewajiban membayar ke anggota yang memiliki piutang.

Bagian ini penting karena:

- Memberikan gambaran akhir keuangan grup
- Menjadi dasar penyelesaian pembayaran antar anggota

2. Detail *Split* per Transaksi

Bagian ini menampilkan rincian pembagian biaya untuk setiap transaksi sehingga *user* dapat melihat bagaimana angka *Net Balance* terbentuk.

- Transaksi T2 – *Even Split*

Total : 500.000,00

Payer : Muti (U1)

Karena menggunakan *Even Split*, maka:

- Total biaya dibagi sama rata ke semua peserta
- Jumlah peserta = 4 orang
- Masing-masing membayar: 125.000,00

Ini cocok untuk transaksi sederhana seperti: patungan makan, bensin, atau tiket dengan porsi yang sama.

- Transaksi T1 – *Uneven Split*

Total : 321.540,00

Payer : Muti (U1)

Karena menggunakan *Uneven Split*, maka:

- Setiap orang membayar sesuai item yang dikonsumsi
- Ditambah biaya tambahan seperti pajak/*service*
- Pembagian tidak sama, tergantung input item dan nominal

Contoh:

- Muti membayar item tertentu + *share* biaya tambahan → 77.285,00
- Alish membayar item berbeda → 98.385,00
- dan seterusnya

Transaksi ini mencerminkan kondisi dunia nyata, misalnya: makan bersama tetapi pesan menu yang berbeda-beda.

S

Splitit

Summary

← Home

Manajemen Grup

Tambah Transaksi

Ringkasan Biaya - Sushi

Positif = piutang, negatif = utang.

Net Balance (Hutang/Piutang)

USER	BALANCE	STATUS
Muti (U1)	619.255,00	Piutang
Alish (U2)	-223.385,00	Utang
Edel (U3)	-194.385,00	Utang
Ann (U4)	-201.485,00	Utang

<div>S Splitit Summary</div>	<div>Detail Split per Transaksi</div> <div>Transaksi: T2 • Total: 500.000,00</div> <div>Payer: Muti (U1)</div> <table> <tr> <th>PARTICIPANT</th><th>SHARE (HARUS BAYAR)</th></tr> <tr> <td>Muti (U1)</td><td>125.000,00</td></tr> <tr> <td>Alish (U2)</td><td>125.000,00</td></tr> <tr> <td>Edel (U3)</td><td>125.000,00</td></tr> <tr> <td>Ann (U4)</td><td>125.000,00</td></tr> </table> <div>Transaksi: T1 • Total: 321.540,00</div> <div>Payer: Muti (U1)</div> <table> <tr> <th>PARTICIPANT</th><th>SHARE (HARUS BAYAR)</th></tr> <tr> <td>Muti (U1)</td><td>77.285,00</td></tr> <tr> <td>Alish (U2)</td><td>98.385,00</td></tr> </table>	PARTICIPANT	SHARE (HARUS BAYAR)	Muti (U1)	125.000,00	Alish (U2)	125.000,00	Edel (U3)	125.000,00	Ann (U4)	125.000,00	PARTICIPANT	SHARE (HARUS BAYAR)	Muti (U1)	77.285,00	Alish (U2)	98.385,00
PARTICIPANT	SHARE (HARUS BAYAR)																
Muti (U1)	125.000,00																
Alish (U2)	125.000,00																
Edel (U3)	125.000,00																
Ann (U4)	125.000,00																
PARTICIPANT	SHARE (HARUS BAYAR)																
Muti (U1)	77.285,00																
Alish (U2)	98.385,00																

5.7. Demo Website

Berikut adalah *link* video demo *website* kelompok kami:

<https://youtu.be/mzvX5mVkewo?si=bNHlx0VFRoBHSMec>

PENUTUP

6.1. Kesimpulan

Aplikasi “SplitIt: *Smart Expense Splitter*” berhasil dikembangkan sebagai solusi untuk membantu pengelolaan dan pembagian pengeluaran dalam suatu kelompok secara adil dan transparan. Aplikasi ini mampu mencatat transaksi, mengelola grup dan anggota, serta menghitung pembagian biaya baik secara rata (*even*) maupun tidak rata (*uneven*) sehingga dapat mengurangi kesalahan perhitungan manual dan potensi konflik antar anggota kelompok.

Dari sisi implementasi teknis, aplikasi SplitIt telah menerapkan konsep Pemrograman Berorientasi Objek (PBO) dengan baik, meliputi *encapsulation*, *abstraction*, *inheritance*, dan *polymorphism*. Penerapan *Strategy Design Pattern* melalui *interface* ISplitStrategy memungkinkan sistem menangani berbagai metode pembagian biaya secara fleksibel tanpa mengubah struktur utama program. Penggunaan *abstract class* BaseEntity juga membantu menjaga konsistensi dan kerapian struktur data dalam aplikasi.

Secara keseluruhan, proyek ini berhasil memenuhi kebutuhan fungsional sebagai aplikasi pembagi tagihan dan menjadi sarana penerapan konsep PBO dalam pengembangan perangkat lunak nyata. Aplikasi SplitIt menunjukkan bahwa perancangan arsitektur yang baik dan modular dapat menghasilkan sistem yang mudah dikembangkan dan dipelihara di masa mendatang.