Expense Management System

1. Scope & Assumptions

This model supports:

- Multi-user system (each user has their own data)
- Multiple **accounts** (cash, bank, card, wallet…)
- **Expenses & incomes** with categories
- **Transfers** between accounts
- **Budgets** and category-wise limits
- **Recurring transactions**
- Tags, attachments, merchants, payment methods
- Multi-currency (optional but included in design)

You can treat some modules as *core* and others as *advanced*.

---

2. Core Domain Entities

2.1 `User`

Represents a registered user.

Field	Type	Notes
id (PK)	BIGINT	Primary key
full_name	VARCHAR(150)	
email	VARCHAR(150)	Unique
password_hash	VARCHAR(255)	Bcrypt/Argon2 etc.
status	ENUM	ACTIVE, INACTIVE, LOCKED, DELETED
preferred_currency	CHAR(3)	FK → Currency.code (e.g., PKR, USD)
created_at	TIMESTAMP	
updated_at	TIMESTAMP	

2.2 `Account`

Wallets, bank accounts, credit cards, etc.

Field	Type	Notes
id (PK)	BIGINT	
user_id (FK)	BIGINT	→ User.id (owner)
name	VARCHAR(100)	e.g., “Cash”, “HBL Current”, “Credit Card”
type	ENUM	CASH, BANK, CREDIT_CARD, MOBILE_WALLET, OTHER
currency_code	CHAR(3)	FK → Currency.code
initial_balance	DECIMAL(18,2)	For opening balance
current_balance	DECIMAL(18,2)	Optional: cached balance for fast display
is_archived	BOOLEAN	Hide inactive accounts
created_at	TIMESTAMP	
updated_at	TIMESTAMP	

Relationship:

- User (1) —— (N) Account

2.3 `Category`

Expense/Income category, supports hierarchy (parent-child).

Field	Type	Notes
id (PK)	BIGINT	
user_id (FK)	BIGINT	Nullable; null = system default category
parent_category_id	BIGINT	Self-FK → Category.id (optional for subcategories)
name	VARCHAR(100)	e.g., “Food”, “Transport”, “Salary”
type	ENUM	EXPENSE, INCOME, TRANSFER
icon	VARCHAR(50)	Optional – for UI
color	VARCHAR(20)	Optional – e.g., hex color
sort_order	INT	Optional – for display ordering
is_archived	BOOLEAN	
created_at	TIMESTAMP	
updated_at	TIMESTAMP	

Relationships:

- User (1) —— (N) Category
- Category (1) —— (N) Category (self-parent for subcategories)

2.4 `Transaction`

Core table for all money events: expenses, incomes, transfers, adjustments.

Field	Type	Notes
id (PK)	BIGINT	
user_id (FK)	BIGINT	→ User.id
account_id (FK)	BIGINT	→ Account.id
category_id (FK)	BIGINT	→ Category.id
payment_method_id (FK)	BIGINT	→ PaymentMethod.id (optional)
merchant_id (FK)	BIGINT	→ Merchant.id (optional)
type	ENUM	EXPENSE, INCOME, TRANSFER_OUT, TRANSFER_IN, ADJUSTMENT
amount	DECIMAL(18,2)	Amount in account's currency
currency_code	CHAR(3)	FK → Currency.code
exchange_rate_to_base	DECIMAL(18,6)	For base currency reporting (optional)
base_amount	DECIMAL(18,2)	amount * exchange_rate_to_base
transaction_date	DATE	The logical date of transaction
transaction_time	TIME	Optional – for detailed logs
status	ENUM	CLEARED, PENDING, VOID
reference_number	VARCHAR(100)	Bank ref, invoice no, etc.
description	VARCHAR(255)	Notes
is_recurring_instance	BOOLEAN	True if generated from RecurringRule
recurring_rule_id (FK)	BIGINT	→ RecurringRule.id (nullable)
linked_transaction_id (FK)	BIGINT	Self-FK → Transaction.id (for transfer pair)
created_at	TIMESTAMP	
updated_at	TIMESTAMP	

Key ideas:

- For **transfers**, create **two** rows:
  - One with type = TRANSFER_OUT (from Account A)
  - One with type = TRANSFER_IN (to Account B)
  - Link them via `linked_transaction_id`.
- For **simple app**, you can ignore `base_amount` and `exchange_rate_to_base`.

Relationships:

- User (1) —— (N) Transaction
- Account (1) —— (N) Transaction
- Category (1) —— (N) Transaction
- PaymentMethod (1) —— (N) Transaction
- Merchant (1) —— (N) Transaction
- RecurringRule (1) —— (N) Transaction (optional)
- Transaction (1) —— (1) Transaction via `linked_transaction_id` for dual-entry transfers

---

3. Supporting / Advanced Entities

3.1 `PaymentMethod`

Describes how the transaction was paid (card, cash, transfer, etc.).

Field	Type	Notes
id (PK)	BIGINT	
user_id (FK)	BIGINT	→ User.id
name	VARCHAR(100)	e.g., “Visa Debit”, “JazzCash”
type	ENUM	CASH, CARD, BANK_TRANSFER, WALLET, OTHER
last4	VARCHAR(4)	For cards (optional)
is_archived	BOOLEAN	
created_at	TIMESTAMP	
updated_at	TIMESTAMP	

3.2 `Merchant`

Saved vendors/shops.

Field	Type	Notes
id (PK)	BIGINT	
user_id (FK)	BIGINT	→ User.id
name	VARCHAR(150)	e.g., “Al-Fateh Store”, “Total Pump”
category_id	BIGINT	Default Category (optional)
address	VARCHAR(255)	Optional
phone	VARCHAR(50)	Optional
website	VARCHAR(150)	Optional
created_at	TIMESTAMP	

Field	Type	Notes
updated_at	TIMESTAMP	

3.3 `Tag` & `TransactionTag` (Many-to-Many)

Free-form labels like “Office”, “Trip to Karachi”.

Tag

Field	Type	Notes
id (PK)	BIGINT	
user_id (FK)	BIGINT	→ User.id
name	VARCHAR(50)	
color	VARCHAR(20)	Optional
created_at	TIMESTAMP	

TransactionTag (junction)

Field	Type	Notes
transaction_id FK	BIGINT	→ Transaction.id (PK part)
tag_id FK	BIGINT	→ Tag.id (PK part)

Composite PK: (transaction_id, tag_id)

3.4 `Attachment`

Store receipts, images, PDFs linked to transactions.

Field	Type	Notes
id (PK)	BIGINT	
transaction_id	BIGINT	→ Transaction.id
file_name	VARCHAR(200)	Original file name
file_path	VARCHAR(255)	Path/URL to storage (local / cloud)
mime_type	VARCHAR(100)	e.g., “image/jpeg”, “application/pdf”
file_size_bytes	BIGINT	Optional
uploaded_at	TIMESTAMP	

3.5 `Budget` & `BudgetItem`

Define budgets per period and per category.

Budget

Field	Type	Notes
id (PK)	BIGINT	
user_id (FK)	BIGINT	→ User.id
name	VARCHAR(100)	e.g., “Monthly – Jan 2026”
period_type	ENUM	MONTHLY, WEEKLY, YEARLY, CUSTOM
start_date	DATE	
end_date	DATE	
total_limit	DECIMAL(18,2)	Optional; global limit
notes	VARCHAR(255)	Optional
created_at	TIMESTAMP	
updated_at	TIMESTAMP	

BudgetItem

Field	Type	Notes
id (PK)	BIGINT	
budget_id (FK)	BIGINT	→ Budget.id
category_id (FK)	BIGINT	→ Category.id (usually EXPENSE categories)
limit_amount	DECIMAL(18,2)	
warning_percent	INT	e.g., 80 means warn at 80% consumption

3.6 `RecurringRule`

Rules for auto-generating repeated transactions (rent, salary, subscriptions).

Field	Type	Notes
id (PK)	BIGINT	
user_id (FK)	BIGINT	→ User.id
base_transaction_id	BIGINT	→ Transaction.id (template)
frequency	ENUM	DAILY, WEEKLY, MONTHLY, YEARLY
interval	INT	e.g., every 1 month, every 2 weeks
day_of_month	INT	For MONTHLY (1–31, handle 30/31 logic in code)
day_of_week	ENUM	MON, TUE, WED, … (for weekly rules)

Field	Type	Notes
start_date	DATE	
end_date	DATE	Nullable – no end
next_run_date	DATE	Scheduler will check and generate transactions from this date
is_active	BOOLEAN	
created_at	TIMESTAMP	
updated_at	TIMESTAMP	

3.7 `Currency` & `ExchangeRate` (Optional but powerful)

Currency

Field	Type	Notes
code (PK)	CHAR(3)	PKR, USD, EUR, etc.
name	VARCHAR(50)	
symbol	VARCHAR(10)	"Rs", "$" etc.
decimal_places	INT	Usually 2

ExchangeRate

Field	Type	Notes
id (PK)	BIGINT	
from_code	CHAR(3)	FK → Currency.code
to_code	CHAR(3)	FK → Currency.code
rate	DECIMAL(18,6)	
rate_date	DATE	
created_at	TIMESTAMP	

You can use this to compute `base_amount` in Transaction for consolidated reporting.

3.8 `NotificationPreference`

For alerts like "budget exceeded", "large expense", etc.

Field	Type	Notes
id (PK)	BIGINT	
user_id (FK)	BIGINT	→ User.id

Field	Type	Notes
type	ENUM	BUDGET_THRESHOLD, LARGE_EXPENSE, RECURRING_FAIL…
channel	ENUM	EMAIL, PUSH, SMS, IN_APP
is_enabled	BOOLEAN	
threshold_amount	DECIMAL(18,2)	For “LARGE_EXPENSE”

3.9 `AuditLog` (Optional but useful in a serious system)

Field	Type	Notes
id (PK)	BIGINT	
user_id (FK)	BIGINT	→ User.id
entity_type	VARCHAR(50)	e.g., “TRANSACTION”, “ACCOUNT”
entity_id	BIGINT	The affected record ID
action	ENUM	CREATE, UPDATE, DELETE, LOGIN, EXPORT
timestamp	TIMESTAMP	
details	JSON / TEXT	Old/new values, IP, etc.

---

4. Relationship Summary (High-Level)

- **User**
  - 1 — N Accounts
  - 1 — N Categories
  - 1 — N Transactions
  - 1 — N PaymentMethods
  - 1 — N Merchants
  - 1 — N Tags
  - 1 — N Budgets
  - 1 — N RecurringRules
  - 1 — N NotificationPreferences
  - 1 — N AuditLogs
- **Account**
  - 1 — N Transactions
- **Category**
  - 1 — N Transactions
  - 1 — N BudgetItems
  - 1 — N Merchants (default category)
  - 1 — N child Categories
- **Transaction**
  - N — N Tags (via TransactionTag)
  - 1 — N Attachments