Bank application ER-Diagram

Banking application ER diagram design with **exactly 10 tables**, covering core retail banking features (customers, accounts, transactions, loans, cards, payees, transfers, and auth).

1. Entities with key attributes
2. Relationships & cardinalities
3. A Mermaid ER diagram you can paste into tools that support Mermaid
4. A clean SQL DDL starter (normalized, relational)

1) Customer

- **PK**: customer_id
- Attributes: first_name, last_name, email (unique), phone, date_of_birth, address_line, city, state, postal_code, country, created_at, status
- Notes: A customer can have multiple accounts, cards, loans, payees.

2) Branch

- **PK**: branch_id
- Attributes: name, ifsc_code/routing_number (unique), address, city, state, postal_code, country, phone
- Notes: Accounts are held at a branch.

3) Account

- **PK**: account_id
- **FKs**: customer_id → Customer, branch_id → Branch
- Attributes: account_number (unique), account_type (Savings/Checking/FD), currency, balance, status, opened_at, closed_at
- Notes: One customer → many accounts.

4) Transaction

- **PK**: transaction_id
- **FK**: account_id → Account
- Attributes: txn_type (credit/debit), amount, currency, posted_at, reference, description, balance_after
- Notes: Many transactions per account.

5) Loan

- **PK**: loan_id
- **FKs**: customer_id → Customer, branch_id → Branch
- Attributes: loan_number (unique), loan_type (Home/Auto/Personal), principal_amount, interest_rate, term_months, start_date, maturity_date, status
- Notes: A customer can have multiple loans.

6) LoanPayment

- **PK**: payment_id
- **FK**: loan_id → Loan
- Attributes: due_date, paid_date, amount_due, amount_paid, penalty_amount, status
- Notes: Installments for loans.

7) Card

- **PK**: card_id
- **FKs**: customer_id → Customer, account_id → Account (for settlement)
- Attributes: card_number (tokenized), card_type (Debit/Credit), network (Visa/Mastercard/RuPay), expiry_month, expiry_year, status, issued_at
- Notes: Cards are linked to customers and typically settle to an account (for debit) or a statement account (for credit).

8) Payee

- **PK**: payee_id
- **FK**: customer_id → Customer
- Attributes: payee_name, payee_account_number, payee_bank_name, payee_ifsc/routing_number, nickname, created_at, status
- Notes: Saved payees for quick transfers/bill payments.

9) Transfer

- **PK**: transfer_id
- **FKs**: from_account_id → Account, to_account_id → Account (can be external or internal)
- Attributes: amount, currency, initiated_at, completed_at, status, channel (Mobile/Web/Branch), reference
- Notes: Represents fund transfers (NEFT/RTGS/IMPS/internal).

10) User

- **PK**: user_id
- **FK**: customer_id → Customer (nullable for staff/ops users if you extend)
- Attributes: username (unique), password_hash, email (unique), phone, role (Customer/Admin/Support), mfa_enabled, last_login_at, status, created_at
- Notes: Authentication/authorization. Keep PII and secrets secure.

---

2) Relationships & Cardinalities

- **Customer 1—N Account**: one customer can hold multiple accounts.
- **Branch 1—N Account**: accounts belong to a branch.
- **Account 1—N Transaction**: each account has many transactions.

- **Customer 1—N Loan**: customer may have multiple loans.
- **Branch 1—N Loan**: loans issued by a branch.
- **Loan 1—N LoanPayment**: installments over time.
- **Customer 1—N Card**: multiple cards per customer.
- **Account 1—N Card**: cards settle to an account (esp. debit).
- **Customer 1—N Payee**: saved beneficiaries per customer.
- **Account 1—N Transfer (from_account)** and **Account 1—N Transfer (to_account)**: transfers link two accounts.
- **Customer 1—1..N User**: typical 1:1 for retail; could be >1 for multiple channels/roles.

4) SQL DDL Starter (PostgreSQL-flavored)

You can adapt types for MySQL/SQL Server. Indexes are added for key lookups and uniqueness.

SQL

```
-- 1) Customer
CREATE TABLE Customer (
customer_id SERIAL PRIMARY KEY,
first_name VARCHAR(100) NOT NULL,
last_name VARCHAR(100) NOT NULL,
email VARCHAR(255) UNIQUE NOT NULL,
phone VARCHAR(30),
date_of_birth DATE,
address_line VARCHAR(255),
city VARCHAR(120),
state VARCHAR(120),
postal_code VARCHAR(20),
country VARCHAR(120),
created_at TIMESTAMP NOT NULL DEFAULT NOW(),
status VARCHAR(30) NOT NULL DEFAULT 'ACTIVE'
);

-- 2) Branch
CREATE TABLE Branch (
branch_id SERIAL PRIMARY KEY,
name VARCHAR(150) NOT NULL,
```

```
ifsc_code VARCHAR(20) UNIQUE NOT NULL,
address VARCHAR(255),
city VARCHAR(120),
state VARCHAR(120),
postal_code VARCHAR(20),
country VARCHAR(120),
phone VARCHAR(30)
);

-- 3) Account
CREATE TABLE Account (
account_id SERIAL PRIMARY KEY,
customer_id INT NOT NULL REFERENCES Customer(customer_id),
branch_id INT NOT NULL REFERENCES Branch(branch_id),
account_number VARCHAR(34) UNIQUE NOT NULL, -- supports IBAN-like lengths
account_type VARCHAR(30) NOT NULL,
currency VARCHAR(3) NOT NULL,
balance NUMERIC(18,2) NOT NULL DEFAULT 0,
status VARCHAR(30) NOT NULL DEFAULT 'OPEN',
opened_at TIMESTAMP NOT NULL DEFAULT NOW(),
closed_at TIMESTAMP
);
CREATE INDEX idx_account_customer ON Account(customer_id);
CREATE INDEX idx_account_branch ON Account(branch_id);

-- 4) Transaction
CREATE TABLE Transaction (
transaction_id BIGSERIAL PRIMARY KEY,
account_id INT NOT NULL REFERENCES Account(account_id),
txn_type VARCHAR(10) NOT NULL CHECK (txn_type IN ('credit','debit')),
amount NUMERIC(18,2) NOT NULL CHECK (amount > 0),
currency VARCHAR(3) NOT NULL,
posted_at TIMESTAMP NOT NULL DEFAULT NOW(),
reference VARCHAR(64),
```

```
description VARCHAR(255),
balance_after NUMERIC(18,2)
);
CREATE INDEX idx_txn_account_posted ON Transaction(account_id, posted_at DESC);

-- 5) Loan
CREATE TABLE Loan (
loan_id SERIAL PRIMARY KEY,
customer_id INT NOT NULL REFERENCES Customer(customer_id),
branch_id INT NOT NULL REFERENCES Branch(branch_id),
loan_number VARCHAR(32) UNIQUE NOT NULL,
loan_type VARCHAR(30) NOT NULL,
principal_amount NUMERIC(18,2) NOT NULL CHECK (principal_amount > 0),
interest_rate NUMERIC(6,4) NOT NULL CHECK (interest_rate >= 0),
term_months INT NOT NULL CHECK (term_months > 0),
start_date DATE NOT NULL,
maturity_date DATE,
status VARCHAR(30) NOT NULL DEFAULT 'ACTIVE'
);
CREATE INDEX idx_loan_customer ON Loan(customer_id);

-- 6) LoanPayment
CREATE TABLE LoanPayment (
payment_id SERIAL PRIMARY KEY,
loan_id INT NOT NULL REFERENCES Loan(loan_id),
due_date DATE NOT NULL,
paid_date DATE,
amount_due NUMERIC(18,2) NOT NULL CHECK (amount_due >= 0),
amount_paid NUMERIC(18,2) CHECK (amount_paid >= 0),
penalty_amount NUMERIC(18,2) DEFAULT 0 CHECK (penalty_amount >= 0),
status VARCHAR(30) NOT NULL DEFAULT 'DUE'
);
CREATE INDEX idx_payment_loan_due ON LoanPayment(loan_id, due_date);
```

Bank application ER-Diagram

```
-- 7) Card
CREATE TABLE Card (
card_id SERIAL PRIMARY KEY,
customer_id INT NOT NULL REFERENCES Customer(customer_id),
account_id INT NOT NULL REFERENCES Account(account_id),
card_number VARCHAR(25) UNIQUE NOT NULL, -- store tokenized PAN, not raw PAN
card_type VARCHAR(20) NOT NULL CHECK (card_type IN ('Debit','Credit')),
network VARCHAR(20) NOT NULL CHECK (network IN ('Visa','Mastercard','RuPay','Amex')),
expiry_month INT NOT NULL CHECK (expiry_month BETWEEN 1 AND 12),
expiry_year INT NOT NULL CHECK (expiry_year >= EXTRACT(YEAR FROM NOW())),
status VARCHAR(30) NOT NULL DEFAULT 'ACTIVE',
issued_at TIMESTAMP NOT NULL DEFAULT NOW()
);
CREATE INDEX idx_card_customer ON Card(customer_id);

-- 8) Payee
CREATE TABLE Payee (
payee_id SERIAL PRIMARY KEY,
customer_id INT NOT NULL REFERENCES Customer(customer_id),
payee_name VARCHAR(150) NOT NULL,
payee_account_number VARCHAR(34) NOT NULL,
payee_bank_name VARCHAR(150),
payee_ifsc VARCHAR(20),
nickname VARCHAR(60),
created_at TIMESTAMP NOT NULL DEFAULT NOW(),
status VARCHAR(30) NOT NULL DEFAULT 'ACTIVE'
);
CREATE INDEX idx_payee_customer ON Payee(customer_id);

-- 9) Transfer
CREATE TABLE Transfer (
transfer_id SERIAL PRIMARY KEY,
from_account_id INT NOT NULL REFERENCES Account(account_id),
to_account_id INT NOT NULL REFERENCES Account(account_id),
```

```
amount NUMERIC(18,2) NOT NULL CHECK (amount > 0),
currency VARCHAR(3) NOT NULL,
initiated_at TIMESTAMP NOT NULL DEFAULT NOW(),
completed_at TIMESTAMP,
status VARCHAR(30) NOT NULL DEFAULT 'PENDING',
channel VARCHAR(20) NOT NULL CHECK (channel IN ('Web','Mobile','Branch','API')),
reference VARCHAR(64)
);
CREATE INDEX idx_transfer_from ON Transfer(from_account_id, initiated_at DESC);
CREATE INDEX idx_transfer_to ON Transfer(to_account_id, initiated_at DESC);

-- 10) User (Authentication)
CREATE TABLE "User" (
user_id SERIAL PRIMARY KEY,
customer_id INT REFERENCES Customer(customer_id),
username VARCHAR(64) UNIQUE NOT NULL,
password_hash VARCHAR(255) NOT NULL,
email VARCHAR(255) UNIQUE NOT NULL,
phone VARCHAR(30),
role VARCHAR(30) NOT NULL CHECK (role IN ('Customer','Admin','Support')),
mfa_enabled BOOLEAN NOT NULL DEFAULT TRUE,
last_login_at TIMESTAMP,
status VARCHAR(30) NOT NULL DEFAULT 'ACTIVE',
created_at TIMESTAMP NOT NULL DEFAULT NOW()
);
```

Show more lines

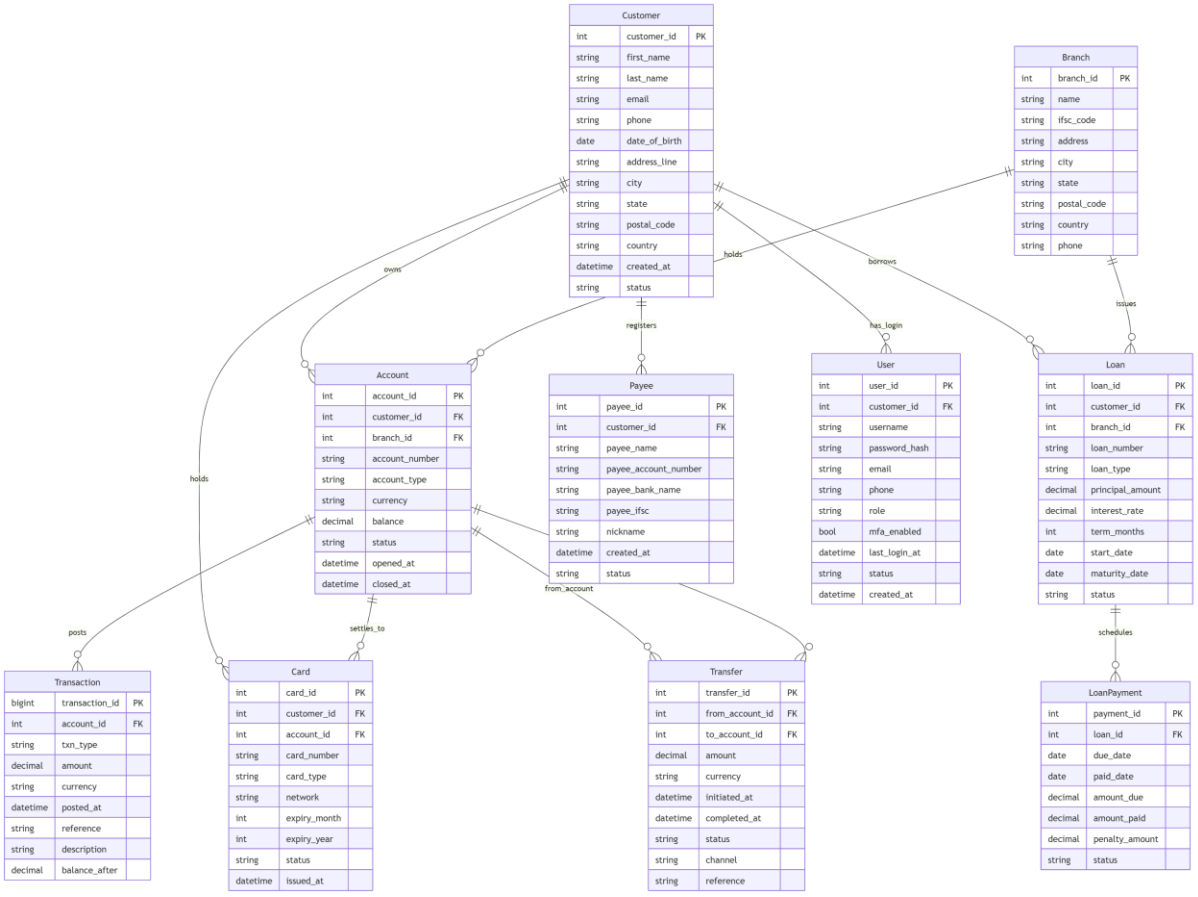
---

5) Design Notes & Best Practices

- **Normalization**: Tables are in 3NF; transactional and reference data are separated.
- **Security & Compliance**:
  - Store **tokenized** card numbers (never raw PAN); enforce encryption at rest for PII.
  - Apply **RBAC** via User.role; use **MFA** and salted hashing (e.g., bcrypt/Argon2).
- **Auditing**: Consider adding AuditLog(user_id, action, resource, created_at, metadata) if you need change tracking.

Bank application ER-Diagram

- **Performance**: Composite indexes on hot paths (e.g., (account_id, posted_at) for statements).
- **Integrity**: Use constraints for amounts, statuses, and referential rules; consider **ON DELETE RESTRICT** on financial FKs.
- **Localization**: Currency as ISO 4217 (VARCHAR(3)), and consider time zone handling (TIMESTAMPTZ).



openapi: 3.0.3

info:

title: Banking API

version: 1.0.0

description: >

REST API for a retail banking application covering Customers, Branches, Accounts,

Transactions, Loans, Loan Payments, Cards, Payees, Transfers, and Users.

servers:

- url: https://api.examplebank.com/v1

Bank application ER-Diagram

```
tags:
  - name: Customers
  - name: Branches
  - name: Accounts
  - name: Transactions
  - name: Loans
  - name: LoanPayments
  - name: Cards
  - name: Payees
  - name: Transfers
  - name: Users
security:
  - bearerAuth: []

paths:
  ############################################################
  # Customers
  ############################################################
  /customers:
    get:
      tags: [Customers]
      summary: List customers
      parameters:
        - $ref: '#/components/parameters/q'
        - $ref: '#/components/parameters/status'
        - $ref: '#/components/parameters/sort'
        - $ref: '#/components/parameters/page'
        - $ref: '#/components/parameters/pageSize'
      responses:
        '200':
          description: List of customers
          headers:
            X-Total-Count:
              schema: { type: integer }
```

Bank application ER-Diagram

```
      content:
        application/json:
          schema:
            type: array
            items: { $ref: '#/components/schemas/Customer' }
  post:
    tags: [Customers]
    summary: Create customer
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/CustomerCreate' }
    responses:
      '201':
        description: Created
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Customer' }

/customers/{customerId}:
  get:
    tags: [Customers]
    summary: Get customer by ID
    parameters:
      - $ref: '#/components/parameters/customerId'
    responses:
      '200':
        description: Customer
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Customer' }
      '404': { $ref: '#/components/responses/NotFound' }
  patch:
```

Bank application ER-Diagram

```
    tags: [Customers]
    summary: Update customer (partial)
    parameters:
     - $ref: '#/components/parameters/customerId'
    requestBody:
     required: true
     content:
      application/json:
       schema: { $ref: '#/components/schemas/CustomerUpdate' }
    responses:
     '200':
      description: Updated
      content:
       application/json:
        schema: { $ref: '#/components/schemas/Customer' }
     '404': { $ref: '#/components/responses/NotFound' }
  delete:
    tags: [Customers]
    summary: Deactivate  customer
    description: Soft-delete via status=INACTIVE
    parameters:
     - $ref: '#/components/parameters/customerId'
    responses:
     '204': { description: Deactivated }
     '404': { $ref: '#/components/responses/NotFound' }


##############################################################
# Branches
##############################################################
/branches:
  get:
    tags: [Branches]
    summary: List branches
    parameters:
```

Bank application ER-Diagram

```
      - $ref: '#/components/parameters/q'
      - $ref: '#/components/parameters/sort'
      - $ref: '#/components/parameters/page'
      - $ref: '#/components/parameters/pageSize'
    responses:
      '200':
        description: List of branches
        content:
          application/json:
            schema:
              type: array
              items: { $ref: '#/components/schemas/Branch' }
  post:
    tags: [Branches]
    summary: Create branch
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/BranchCreate' }
    responses:
      '201':
        description: Created
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Branch' }

/branches/{branchId}:
  get:
    tags: [Branches]
    summary: Get branch
    parameters:
      - $ref: '#/components/parameters/branchId'
    responses:
```

Bank application ER-Diagram

```
      '200':
       description: Branch
       content:
        application/json:
         schema: { $ref: '#/components/schemas/Branch' }
      '404': { $ref: '#/components/responses/NotFound' }
    patch:
     tags: [Branches]
     summary: Update branch
     parameters:
      - $ref: '#/components/parameters/branchId'
     requestBody:
      required: true
      content:
       application/json:
        schema: { $ref: '#/components/schemas/BranchUpdate' }
     responses:
      '200':
       description: Updated
       content:
        application/json:
         schema: { $ref: '#/components/schemas/Branch' }
      '404': { $ref: '#/components/responses/NotFound' }


  ############################################################
  # Accounts
  ############################################################
  /accounts:
   get:
    tags: [Accounts]
    summary: List accounts
    parameters:
     - in: query
       name: customerId
```

Bank application ER-Diagram

```
      schema: { type: integer }
    - in: query
      name: branchId
      schema: { type: integer }
    - $ref: '#/components/parameters/status'
    - $ref: '#/components/parameters/sort'
    - $ref: '#/components/parameters/page'
    - $ref: '#/components/parameters/pageSize'
  responses:
    '200':
      description: List of accounts
      content:
        application/json:
          schema:
            type: array
            items: { $ref: '#/components/schemas/Account' }
  post:
    tags: [Accounts]
    summary: Open account
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/AccountCreate' }
    responses:
      '201':
        description: Created
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Account' }

/accounts/{accountId}:
  get:
    tags: [Accounts]
```

Bank application ER-Diagram

```
    summary: Get account
    parameters:
      - $ref: '#/components/parameters/accountId'
    responses:
      '200':
        description: Account
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Account' }
      '404': { $ref: '#/components/responses/NotFound' }
  patch:
    tags: [Accounts]
    summary: Update account (non-balance fields)
    parameters:
      - $ref: '#/components/parameters/accountId'
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/AccountUpdate' }
    responses:
      '200':
        description: Updated
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Account' }
  delete:
    tags: [Accounts]
    summary: Close account
    parameters:
      - $ref: '#/components/parameters/accountId'
    responses:
      '204': { description: Closed }
      '409': { description: Cannot close non-zero balance }
```

Bank application ER-Diagram

```
/accounts/{accountId}/transactions:
  get:
    tags: [Transactions]
    summary: List transactions for account
    parameters:
      - $ref: '#/components/parameters/accountId'
      - in: query
        name: from
        description: ISO date-time from
        schema: { type: string, format: date-time }
      - in: query
        name: to
        description: ISO date-time to
        schema: { type: string, format: date-time }
      - $ref: '#/components/parameters/sort'
      - $ref: '#/components/parameters/page'
      - $ref: '#/components/parameters/pageSize'
    responses:
      '200':
        description: Transactions
        content:
          application/json:
            schema:
              type: array
              items: { $ref: '#/components/schemas/Transaction' }
  post:
    tags: [Transactions]
    summary: Post a transaction (credit/debit)
    description: >
      Creates a ledger entry. Balance is computed server-side. Use Idempotency-Key.
    parameters:
      - $ref: '#/components/parameters/accountId'
      - $ref: '#/components/parameters/IdempotencyKey'
```

Bank application ER-Diagram

```
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/TransactionPost' }
    responses:
      '201':
        description: Created
        headers:
          Idempotency-Replayed:
            schema: { type: boolean }
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Transaction' }

  /transactions/{transactionId}:
    get:
      tags: [Transactions]
      summary: Get transaction
      parameters:
        - $ref: '#/components/parameters/transactionId'
      responses:
        '200':
          description: Transaction
          content:
            application/json:
              schema: { $ref: '#/components/schemas/Transaction' }
        '404': { $ref: '#/components/responses/NotFound' }

  ############################################################
  # Loans & Loan Payments
  ############################################################
  /loans:
    get:
```

Bank application ER-Diagram

```
    tags: [Loans]
    summary: List loans
    parameters:
      - in: query
        name: customerId
        schema: { type: integer }
      - in: query
        name: branchId
        schema: { type: integer }
      - $ref: '#/components/parameters/status'
      - $ref: '#/components/parameters/sort'
      - $ref: '#/components/parameters/page'
      - $ref: '#/components/parameters/pageSize'
    responses:
      '200':
        description: List of loans
        content:
          application/json:
            schema:
              type: array
              items: { $ref: '#/components/schemas/Loan' }
  post:
    tags: [Loans]
    summary: Create loan
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/LoanCreate' }
    responses:
      '201':
        description: Created
        content:
          application/json:
```

```
          schema: { $ref: '#/components/schemas/Loan' }


 /loans/{loanId}:
  get:
   tags: [Loans]
   summary: Get loan
   parameters:
    - $ref: '#/components/parameters/loanId'
   responses:
    '200':
     description: Loan
     content:
      application/json:
       schema: { $ref: '#/components/schemas/Loan' }
    '404': { $ref: '#/components/responses/NotFound' }
  patch:
   tags: [Loans]
   summary: Update loan (status, maturity_date)
   parameters:
    - $ref: '#/components/parameters/loanId'
   requestBody:
    required: true
    content:
     application/json:
      schema: { $ref: '#/components/schemas/LoanUpdate' }
   responses:
    '200':
     description: Updated
     content:
      application/json:
       schema: { $ref: '#/components/schemas/Loan' }


 /loans/{loanId}/payments:
  get:
```

Bank application ER-Diagram

```
    tags: [LoanPayments]
    summary: List loan payments (schedule & actuals)
    parameters:
      - $ref: '#/components/parameters/loanId'
      - $ref: '#/components/parameters/page'
      - $ref: '#/components/parameters/pageSize'
    responses:
      '200':
        description: Payments
        content:
          application/json:
            schema:
              type: array
              items: { $ref: '#/components/schemas/LoanPayment' }
  post:
    tags: [LoanPayments]
    summary: Record/allocate a payment
    parameters:
      - $ref: '#/components/parameters/loanId'
      - $ref: '#/components/parameters/IdempotencyKey'
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/LoanPaymentPost' }
    responses:
      '201':
        description: Created
        content:
          application/json:
            schema: { $ref: '#/components/schemas/LoanPayment' }

/loan-payments/{paymentId}:
  get:
```

Bank application ER-Diagram

```
    tags: [LoanPayments]
    summary: Get a loan payment
    parameters:
     - $ref: '#/components/parameters/paymentId'
    responses:
     '200':
      description: Payment
      content:
       application/json:
        schema: { $ref: '#/components/schemas/LoanPayment' }
     '404': { $ref: '#/components/responses/NotFound' }


  ############################################################
  # Cards
  ############################################################
  /cards:
   get:
    tags: [Cards]
    summary: List cards
    parameters:
     - in: query
      name: customerId
      schema: { type: integer }
     - in: query
      name: accountId
      schema: { type: integer }
     - $ref: '#/components/parameters/status'
     - $ref: '#/components/parameters/page'
     - $ref: '#/components/parameters/pageSize'
    responses:
     '200':
      description: List of cards
      content:
       application/json:
```

Bank application ER-Diagram

```
          schema:
           type: array
           items: { $ref: '#/components/schemas/Card' }
    post:
     tags: [Cards]
     summary: Issue card
     requestBody:
      required: true
      content:
       application/json:
        schema: { $ref: '#/components/schemas/CardCreate' }
     responses:
      '201':
       description: Created
       content:
        application/json:
         schema: { $ref: '#/components/schemas/Card' }

  /cards/{cardId}:
    get:
     tags: [Cards]
     summary: Get card
     parameters:
      - $ref: '#/components/parameters/cardId'
     responses:
      '200':
       description: Card
       content:
        application/json:
         schema: { $ref: '#/components/schemas/Card' }
      '404': { $ref: '#/components/responses/NotFound' }
    patch:
     tags: [Cards]
     summary: Update card (status, replacement, reissue)
```

Bank application ER-Diagram

```
    parameters:
     - $ref: '#/components/parameters/cardId'
    requestBody:
     required: true
     content:
      application/json:
       schema: { $ref: '#/components/schemas/CardUpdate' }
    responses:
     '200':
      description: Updated
      content:
       application/json:
        schema: { $ref: '#/components/schemas/Card' }


  ############################################################
  # Payees
  ############################################################
  /payees:
   get:
    tags: [Payees]
    summary: List payees
    parameters:
     - in: query
       name: customerId
       required: true
       schema: { type: integer }
     - $ref: '#/components/parameters/page'
     - $ref: '#/components/parameters/pageSize'
    responses:
     '200':
      description: Payees
      content:
       application/json:
        schema:
```

Bank application ER-Diagram

```
          type: array
          items: { $ref: '#/components/schemas/Payee' }
    post:
     tags: [Payees]
     summary: Create payee
     requestBody:
      required: true
      content:
       application/json:
        schema: { $ref: '#/components/schemas/PayeeCreate' }
     responses:
      '201':
       description: Created
       content:
        application/json:
         schema: { $ref: '#/components/schemas/Payee' }

  /payees/{payeeId}:
   get:
    tags: [Payees]
    summary: Get payee
    parameters:
     - $ref: '#/components/parameters/payeeId'
    responses:
     '200':
      description: Payee
      content:
       application/json:
        schema: { $ref: '#/components/schemas/Payee' }
     '404': { $ref: '#/components/responses/NotFound' }
   patch:
    tags: [Payees]
    summary: Update payee (nickname, status)
    parameters:
```

Bank application ER-Diagram

```
    - $ref: '#/components/parameters/payeeId'
  requestBody:
    required: true
    content:
      application/json:
        schema: { $ref: '#/components/schemas/PayeeUpdate' }
  responses:
    '200':
      description: Updated
      content:
        application/json:
          schema: { $ref: '#/components/schemas/Payee' }
 delete:
  tags: [Payees]
  summary: Delete payee
  parameters:
    - $ref: '#/components/parameters/payeeId'
  responses:
    '204': { description: Deleted }


############################################################
# Transfers
############################################################
/transfers:
 get:
  tags: [Transfers]
  summary: List transfers
  parameters:
    - in: query
      name: fromAccountId
      schema: { type: integer }
    - in: query
      name: toAccountId
      schema: { type: integer }
```

Bank application ER-Diagram

```
    - $ref: '#/components/parameters/status'
    - $ref: '#/components/parameters/page'
    - $ref: '#/components/parameters/pageSize'
  responses:
    '200':
      description: Transfers
      content:
        application/json:
          schema:
            type: array
            items: { $ref: '#/components/schemas/Transfer' }
  post:
    tags: [Transfers]
    summary: Initiate transfer (internal/external)
    description: >
      Creates a transfer and enqueues settlement. Use Idempotency-Key.
      On success, ledger entries are created atomically.
    parameters:
      - $ref: '#/components/parameters/IdempotencyKey'
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/TransferCreate' }
    responses:
      '201':
        description: Created
        headers:
          Idempotency-Replayed:
            schema: { type: boolean }
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Transfer' }
```

Bank application ER-Diagram

```
/transfers/{transferId}:
  get:
    tags: [Transfers]
    summary: Get transfer
    parameters:
      - $ref: '#/components/parameters/transferId'
    responses:
      '200':
        description: Transfer
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Transfer' }
      '404': { $ref: '#/components/responses/NotFound' }
  post:
    tags: [Transfers]
    summary: Cancel transfer (if still pending)
    operationId: cancelTransfer
    parameters:
      - $ref: '#/components/parameters/transferId'
    requestBody:
      required: false
    responses:
      '200':
        description: Cancelled
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Transfer' }
      '409':
        description: Cannot cancel non-pending transfer

############################################################
# Users (AuthN/AuthZ identity – **not** customers)
############################################################
/users:
```

Bank application ER-Diagram

```
  get:
    tags: [Users]
    summary: List users
    parameters:
      - in: query
        name: customerId
        schema: { type: integer }
      - in: query
        name: role
        schema:
          type: string
          enum: [Customer, Admin, Support]
      - $ref: '#/components/parameters/status'
      - $ref: '#/components/parameters/page'
      - $ref: '#/components/parameters/pageSize'
    responses:
      '200':
        description: Users
        content:
          application/json:
            schema:
              type: array
              items: { $ref: '#/components/schemas/User' }
  post:
    tags: [Users]
    summary: Create user (login identity)
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/UserCreate' }
    responses:
      '201':
        description: Created
```

```
      content:
        application/json:
          schema: { $ref: '#/components/schemas/User' }


/users/{userId}:
  get:
    tags: [Users]
    summary: Get user
    parameters:
      - $ref: '#/components/parameters/userId'
    responses:
      '200':
        description: User
        content:
          application/json:
            schema: { $ref: '#/components/schemas/User' }
      '404': { $ref: '#/components/responses/NotFound' }
  patch:
    tags: [Users]
    summary: Update user (role, status, mfa)
    parameters:
      - $ref: '#/components/parameters/userId'
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/UserUpdate' }
    responses:
      '200':
        description: Updated
        content:
          application/json:
            schema: { $ref: '#/components/schemas/User' }
```

Bank application ER-Diagram

```
components:
 securitySchemes:
  bearerAuth:
   type: http
   scheme: bearer
   bearerFormat: JWT

 parameters:
  customerId:
   in: path
   name: customerId
   required: true
   schema: { type: integer }
  branchId:
   in: path
   name: branchId
   required: true
   schema: { type: integer }
  accountId:
   in: path
   name: accountId
   required: true
   schema: { type: integer }
  transactionId:
   in: path
   name: transactionId
   required: true
   schema: { type: integer }
  loanId:
   in: path
   name: loanId
   required: true
   schema: { type: integer }
  paymentId:
```

Bank application ER-Diagram

```
    in: path
    name: paymentId
    required: true
    schema: { type: integer }
  cardId:
    in: path
    name: cardId
    required: true
    schema: { type: integer }
  payeeId:
    in: path
    name: payeeId
    required: true
    schema: { type: integer }
  transferId:
    in: path
    name: transferId
    required: true
    schema: { type: integer }
  userId:
    in: path
    name: userId
    required: true
    schema: { type: integer }
  q:
    in: query
    name: q
    description: Free-text search
    schema: { type: string }
  sort:
    in: query
    name: sort
    description: Sort by field, optionally desc (e.g., "created_at,-last_name")
    schema: { type: string }
```

Bank application ER-Diagram

```
  page:
    in: query
    name: page
    schema: { type: integer, minimum: 1, default: 1 }
  pageSize:
    in: query
    name: pageSize
    schema: { type: integer, minimum: 1, maximum: 200, default: 25 }
  status:
    in: query
    name: status
    schema: { type: string }
  IdempotencyKey:
    in: header
    name: Idempotency-Key
    required: false
    description: Provide for POST of financial operations to avoid duplicates
    schema: { type: string, maxLength: 64 }

responses:
  NotFound:
    description: Resource not found
    content:
      application/json:
        schema: { $ref: '#/components/schemas/Error' }

schemas:
  ########################################
  # Core Models
  ########################################
  Customer:
    type: object
    required: [customer_id, first_name, last_name, email, status, created_at]
    properties:
```

Bank application ER-Diagram

customer_id: { type: integer }

first_name: { type: string, maxLength: 100 }

last_name: { type: string, maxLength: 100 }

email: { type: string, format: email }

phone: { type: string, maxLength: 30 }

date_of_birth: { type: string, format: date }

address_line: { type: string }

city: { type: string }

state: { type: string }

postal_code: { type: string }

country: { type: string }

status: { type: string, enum: [ACTIVE, INACTIVE, BLOCKED] }

created_at: { type: string, format: date-time }

CustomerCreate:

type: object

required: [first_name, last_name, email]

properties:

first_name: { type: string }

last_name: { type: string }

email: { type: string, format: email }

phone: { type: string }

date_of_birth: { type: string, format: date }

address_line: { type: string }

city: { type: string }

Domain Behavior & Best Practices

- **Immutability of Ledger**
    - POST /accounts/{id}/transactions **creates** a transaction; no PATCH/DELETE on transactions.
    - POST /transfers performs **debit + credit** atomically; use **Idempotency-Key** to avoid duplicates.
- **Idempotency**
    - Provide Idempotency-Key for financial POSTs: transfers, transactions, loan payments. Server should return 201 on first run and 200 with Idempotency-Replayed: true header on safe replays.
- **Pagination & Sorting**
    - page, pageSize, and sort supported broadly; return X-Total-Count for collection endpoints.
- **Auth & Roles**

- **Bearer JWT**: authorize access by User.role. Example: Customer may only access their own resources; Admin can manage branches, etc.

- **Validation Highlights**
  - Positive amounts; ISO 4217 currency; IFSC format on payees; card PAN must be **tokenized**, never raw.

Quick Start (cURL)

Replace TOKEN with a valid bearer token.

1) Create a Customer

Shell

curl -sS -X POST https://api.examplebank.com/v1/customers \

-H "Authorization: Bearer TOKEN" -H "Content-Type: application/json" \

-d '{

"first_name":"Asha","last_name":"Rao",

"email":"asha.rao@example.com","phone":"+91-9000000000",

"address_line":"12 Gandhi Rd","city":"Kanchipuram","state":"TN","country":"IN"

}'

Show more lines

2) Open an Account

Shell

curl -sS -X POST https://api.examplebank.com/v1/accounts \

-H "Authorization: Bearer TOKEN" -H "Content-Type: application/json" \

-d '{

"customer_id": 1,

"branch_id": 1,

"account_type": "Savings",

"currency": "INR",

"initial_deposit": 2000.00

}'

Show more lines

3) Add a Payee

Shell

curl -sS -X POST https://api.examplebank.com/v1/payees \

-H "Authorization: Bearer TOKEN" -H "Content-Type: application/json" \

-d '{

"customer_id": 1,

"payee_name": "V Kumar",

"payee_account_number": "123456789012",

"payee_bank_name": "Example Bank",

"payee_ifsc": "EXAMP0000123",

"nickname": "Vijay"

}'

4) Transfer Funds (Idempotent)

Shell

curl -sS -X POST https://api.examplebank.com/v1/transfers \

-H "Authorization: Bearer TOKEN" -H "Content-Type: application/json" \

-H "Idempotency-Key: 9f9f9f-20260129-001" \

-d '{

"from_account_id": 1001,

"to_account_id": 2002,

"amount": 500.00,

"currency": "INR",

"channel": "Mobile",

"reference": "Rent-Jan"

}'

Show more lines

5) Post a Manual Transaction

Shell

curl -sS -X POST https://api.examplebank.com/v1/accounts/1001/transactions \

-H "Authorization: Bearer TOKEN" -H "Content-Type: application/json" \

-H "Idempotency-Key: 9f9f9f-20260129-002" \

-d '{"txn_type":"credit","amount":250.00,"currency":"INR","reference":"CASH-DEP"}'

Show more lines
