Internal Payment Management System

CodeLogic:

1. Overview

The Internal Payment Management System is designed to manage users, payments, and financial reporting. It supports role-based access control with three roles:

Admin → Manage users and audit logs
Finance Manager → Manage payments and generate reports
Viewer → View reports

Access to system features is strictly controlled based on user roles.

2. User Roles and Responsibilities

2.1 Admin

Operations:

- Create new user
- Update role of existing user
- Delete (deactivate) user
- View audit logs

Entity Access: User table
Attributes: userId, name, email, password, isActive

Deletion is logical (mark isActive=false) to maintain referential integrity with payment records.

Business Rules:

Password is auto-generated by reversing the username.
Default isActive=true when a user is created.
New user creation first checks if email already exists:
If exists and isActive=false → reactivate user.
If not exists → create new user.

Role must be stored in lowercase.
Updates and deletions are done using email (not userId).

2.2 Finance Manager

Operations:

Add new payment
Update payment status
Generate reports (monthly, quarterly)

Payment Entity:

Attributes: paymentId, amount, type, categoryId, status, managerId, date
Business Rules:

Type must be in Title Case (Incoming, Outgoing)
Status must be in Title Case (Pending, Completed)
Payment requires valid categoryId (1=Salary, 2=Vendor, 3=Client Invoice).
To update status, finance manager must know the paymentId.

Reports:

Monthly & Quarterly reports show total money in each category.

2.3 Viewer

Operations:

View monthly reports
View quarterly reports

Restrictions:

No access to create/update operations.
Read-only access.

3. System Workflows

3.1 Login Workflow

User enters email and password.
System validates credentials.

If valid → role-based menu is displayed.
If invalid → system asks user to type "Yes" (Title case) to retry.
Any other input → system exits.

After logout, system again prompts "Yes" to continue or exit.

3.2 Admin Workflow

Admin logs in.
Menu options: Create User, Update Role, Delete User, View Audit Log.
User CRUD operations follow business rules (password generation, email-based updates, soft deletion).

3.3 Finance Manager Workflow

Finance Manager logs in.
Menu options: Add Payment, Update Payment Status, Generate Report.
Report generation outputs aggregated totals per category for monthly/quarterly.

3.4 Viewer Workflow

Viewer logs in.
Menu options: View Monthly Report, View Quarterly Report.
Read-only access to reports.

4. Data Integrity Rules

User Table must always contain at least 1 Admin.
Default Admin:(In my local database)
name=Mrudula
email=mrudula@gmail.com
password=12345678

Logical deletion using isActive=false.

Enforced casing:
Role = lowercase
Type/Status = Title Case

5. Running Instructions

Ensure database has User table initialized with default Admin.
Start the application → Login screen appears.
Enter valid credentials.
Role-specific menu will be displayed.

Logout returns to main login prompt.

Technical Requirements:

Postgresql,Tables present in postgresql - users,payments,categories(refer to ER diagram for the exact column names and data type), JDK.

NOTE : New User cannot register into the system only admin can add the users

GitHub Link :
https://github.com/Lakshmi-mrudula-dolly/Internal_Payment_Management_System

SETUP INSTRUCTIONS(README .md file) :

Internal Payment Management System
Overview
The Internal Payment Management System is a role-based application that allows organizations to manage users, payments, and financial reports.
It supports three types of users:
Admin → Manage users and audit logs
Finance Manager → Manage payments and generate reports
Viewer → View reports only
The system ensures data integrity, traceability, and clean separation of concerns using DAO, DTO, and Service layers.

Technologies Used
Java 17+
JDBC for database connectivity
PostgreSQL/MySQL as the database
PlantUML for UML diagrams
Maven
GitHub for version control

Project Structure
PaymentManagementSystem/
├── src/
│ ├── tech/zeta/model/ # Entity classes (User, Payment, Category)

```
│   ├── tech/zeta/dao/          # DAO Interfaces
│   ├── tech/zeta/dao/impl/     # DAO Implementations
│   ├── tech/zeta/service/      # Service classes (UserService, PaymentService)
│   ├── tech/zeta/menu/         # Menu-driven UI (AdminMenu, FinanceManagerMenu,
ViewerMenu)
│   └── tech/zeta/util/         # DBUtil for database connection
├── resources/
│   └── schema.sql              # SQL script to create tables (optional)
|   |___ application.properties
```

Database Schema

Users Table
userId (PK, BIGINT)
name (VARCHAR)
email (VARCHAR, unique)
password (VARCHAR)
role (VARCHAR) → lowercase
isActive (BOOLEAN)

Payments Table
paymentId (PK, BIGINT)
amount (DECIMAL)
type (VARCHAR, Title Case: Incoming/Outgoing)
categoryId (FK → Categories)
status (VARCHAR, Title Case: Pending/Completed)
managerId (FK → Users)
date (DATE)

Categories Table
categoryId (PK, INT)
categoryName (VARCHAR) → e.g., Salary, Vendor, Client Invoice

How to Run

Clone the repository:
https://github.com/Lakshmi-mrudula-dolly/Internal_Payment_Management_System/

git clone
cd PaymentManagementSystem

Set up the database:
Create a PostgreSQL/MySQL database (e.g., paymentdb).
Run the SQL script (resources/schema.sql) or manually create tables.
Update application.properties with your database credentials:

```
private static final String URL = "jdbc:postgresql://localhost:5432/paymentdb";
private static final String USER = "your-username";
private static final String PASSWORD = "your-password";
```

Compile and run the project:

```
javac -d bin src/**/*.java
java -cp bin tech.zeta.Main
```

Login with default admin:
Email: mrudula@gmail.com
Password: 12345678

User Roles & Access
Admin → Create, Update, Delete Users; View Audit Log
Finance Manager → Add Payment, Update Status, Generate Reports
Viewer → View Reports only

Reports
Monthly Report → Payments summary by category for the given month
Quarterly Report → Payments summary by category for the given quarter