bank management system mysql database

Here's a detailed **MySQL database design for a Bank Management System**, covering the core aspects such as customer information, accounts, transactions, and loan management.

Database Schema for Bank Management System

1. Customers Table

Stores customer personal details.

```
CREATE TABLE customers (
    customer_id INT PRIMARY KEY AUTO_INCREMENT, -- Unique customer ID
    name VARCHAR(100) NOT NULL,              -- Customer's name
    address VARCHAR(255) NOT NULL,           -- Customer's address
    email VARCHAR(100) UNIQUE NOT NULL,      -- Customer's email
    phone VARCHAR(15) UNIQUE NOT NULL,       -- Customer's phone number
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP -- Account creation timestamp
);
```

2. Accounts Table

Stores account details, such as account types and balances.

```
CREATE TABLE accounts (
    account_number INT PRIMARY KEY AUTO_INCREMENT, -- Unique account number
    customer_id INT NOT NULL,                  -- Foreign key to customers
    account_type ENUM('Savings', 'Current') NOT NULL, -- Type of account
    balance DECIMAL(10, 2) DEFAULT 0.00,       -- Current account balance
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Account creation timestamp
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
);
```

3. Transactions Table

Logs all transactions for auditing and history purposes.

```
CREATE TABLE transactions (
    transaction_id INT PRIMARY KEY AUTO_INCREMENT,  -- Unique transaction ID
    account_number INT NOT NULL,                    -- Foreign key to accounts
    transaction_type ENUM('Deposit', 'Withdrawal', 'Transfer') NOT NULL, -- Type of transaction
    amount DECIMAL(10, 2) NOT NULL,                 -- Transaction amount
    recipient_account INT,                          -- Recipient account for transfers
    transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Date of transaction
    status ENUM('Success', 'Failed') DEFAULT 'Success', -- Status of the transaction
    FOREIGN KEY (account_number) REFERENCES accounts(account_number) ON DELETE CASCADE
);
```

4. Admins Table

Stores admin login credentials and details

```
CREATE TABLE admins (
    admin_id INT PRIMARY KEY AUTO_INCREMENT,        -- Unique admin ID
    username VARCHAR(50) UNIQUE NOT NULL,           -- Admin username
    password VARCHAR(255) NOT NULL,                 -- Encrypted admin password
    email VARCHAR(100) UNIQUE NOT NULL,             -- Admin email
    phone VARCHAR(15) UNIQUE NOT NULL               -- Admin phone number
);
```

5. Loans Table

Tracks customer loan details.

```
CREATE TABLE loans (
    loan_id INT PRIMARY KEY AUTO_INCREMENT,         -- Unique loan ID
```

```
    customer_id INT NOT NULL,                    -- Foreign key to customers
    loan_type ENUM('Home', 'Car', 'Personal', 'Education') NOT NULL, -- Type of loan
    loan_amount DECIMAL(15, 2) NOT NULL,         -- Loan amount
    interest_rate DECIMAL(5, 2) NOT NULL,        -- Interest rate
    loan_start_date DATE NOT NULL,               -- Loan issuance date
    loan_end_date DATE NOT NULL,                 -- Loan repayment due date
    status ENUM('Active', 'Closed') DEFAULT 'Active', -- Loan status
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
);
```

6. Login Logs Table

Logs user login attempts for security and auditing purposes.

```
CREATE TABLE login_logs (
    log_id INT PRIMARY KEY AUTO_INCREMENT,       -- Unique log ID
    user_type ENUM('Customer', 'Admin') NOT NULL,  -- Indicates the type of user
    user_id INT NOT NULL,                        -- Refers to customer_id or admin_id
    login_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Timestamp of login
    status ENUM('Success', 'Failed') NOT NULL    -- Login status
);
```

Relationships Between Tables

1. **customers ↔ accounts:**
    - A customer can have multiple accounts (1-to-many relationship).
2. **accounts ↔ transactions:**
    - Each transaction is linked to one account (1-to-many relationship).
3. **customers ↔ loans:**
    - A customer can have multiple loans (1-to-many relationship).
4. **admins ↔ login_logs:**
    - user_id in login_logs references admin_id for admin login attempts.
5. **customers ↔ login_logs:**
    - user_id in login_logs references customer_id for customer login attempts.

SQL Script to Create All Tables

Here's the complete script to create the database and tables:

sql

Copy code

```
-- Create Database
CREATE DATABASE BankManagementSystem;

-- Switch to Database
USE BankManagementSystem;

-- Customers Table
CREATE TABLE customers (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    address VARCHAR(255) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(15) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Accounts Table
CREATE TABLE accounts (
    account_number INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT NOT NULL,
    account_type ENUM('Savings', 'Current') NOT NULL,
    balance DECIMAL(10, 2) DEFAULT 0.00,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
);
```

```
-- Transactions Table
CREATE TABLE transactions (
    transaction_id INT PRIMARY KEY AUTO_INCREMENT,
    account_number INT NOT NULL,
    transaction_type ENUM('Deposit', 'Withdrawal', 'Transfer') NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    recipient_account INT,
    transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status ENUM('Success', 'Failed') DEFAULT 'Success',
    FOREIGN KEY (account_number) REFERENCES accounts(account_number) ON DELETE CASCADE
);

-- Admins Table
CREATE TABLE admins (
    admin_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(15) UNIQUE NOT NULL
);

-- Loans Table
CREATE TABLE loans (
    loan_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT NOT NULL,
    loan_type ENUM('Home', 'Car', 'Personal', 'Education') NOT NULL,
    loan_amount DECIMAL(15, 2) NOT NULL,
    interest_rate DECIMAL(5, 2) NOT NULL,
    loan_start_date DATE NOT NULL,
    loan_end_date DATE NOT NULL,
    status ENUM('Active', 'Closed') DEFAULT 'Active',
```

```
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
);

-- Login Logs Table
CREATE TABLE login_logs (
    log_id INT PRIMARY KEY AUTO_INCREMENT,
    user_type ENUM('Customer', 'Admin') NOT NULL,
    user_id INT NOT NULL,
    login_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status ENUM('Success', 'Failed') NOT NULL
);
```

Testing the Database

1. **Populate tables with test data**:

```
sql
Copy code
INSERT INTO customers (name, address, email, phone)
VALUES ('John Doe', '123 Main St', 'john@example.com', '1234567890');
```

2. **Query tables**:

```
sql
Copy code
SELECT * FROM customers;
```

3. **Use JOINs to verify relationships**:

```
sql
Copy code
SELECT c.name, a.account_type, a.balance
FROM customers c
JOIN accounts a ON c.customer_id = a.customer_id;
```

Let me know if you need assistance integrating this database with your application!