# Bankify – A Scalable Java Banking Backend System

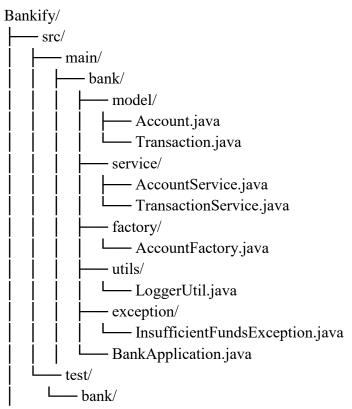
## **Project Overview:**

This project is a simulation of a backend banking system developed in Java 17. It includes object-oriented design, multithreading, custom exceptions, design patterns, and simulated RESTful operations. Ideal for showcasing technical skills to top tech companies.

#### **Tech Stack:**

- Java 17
- Log4j2
- JUnit5
- Maven
- Object-Oriented Programming
- RESTful Simulation

#### **Folder Structure:**



```
service/
AccountServiceTest.java
```

### Account.java

```
package bank.model;
import java.util.UUID;
import java.util.concurrent.locks.ReentrantLock;
public class Account {
  private final String id;
  private String holderName;
  private double balance;
  private final ReentrantLock lock = new ReentrantLock();
  public Account(String holderName, double initialBalance) {
    this.id = UUID.randomUUID().toString();
    this.holderName = holderName;
    this.balance = initialBalance:
  }
  public String getId() { return id; }
  public String getHolderName() { return holderName; }
  public void setHolderName(String holderName) { this.holderName = holderName; }
  public double getBalance() { return balance; }
  public ReentrantLock getLock() { return lock; }
  public void deposit(double amount) {
    if (amount <= 0) throw new IllegalArgumentException("Deposit must be positive.");
    balance += amount;
  }
  public void withdraw(double amount) {
    if (amount <= 0) throw new IllegalArgumentException("Withdrawal must be
positive.");
    if (amount > balance) throw new RuntimeException("Insufficient funds.");
    balance -= amount;
  }
```

```
}
```

## Transaction.java

```
package bank.model;
import java.time.LocalDateTime;
import java.util.UUID;
public class Transaction {
  private final String id;
  private final String fromAccountId;
  private final String to AccountId;
  private final double amount;
  private final LocalDateTime timestamp;
  public Transaction(String fromAccountId, String toAccountId, double amount) {
    this.id = UUID.randomUUID().toString();
    this.fromAccountId = fromAccountId;
    this.toAccountId = toAccountId;
    this.amount = amount;
    this.timestamp = LocalDateTime.now();
  }
  public String getId() { return id; }
  public String getFromAccountId() { return fromAccountId; }
  public String getToAccountId() { return toAccountId; }
  public double getAmount() { return amount; }
  public LocalDateTime getTimestamp() { return timestamp; }
}
```