IntelliBank – Intelligent Transaction System

IntelliBankApplication.java

```
package com.intellibank;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class IntelliBankApplication {
    public static void main(String[] args) {
        SpringApplication.run(IntelliBankApplication.class, args);
    }
}
```

TransactionType.java

```
package com.intellibank.domain;

public enum TransactionType {
    DEPOSIT, WITHDRAWAL, TRANSFER;

    public boolean isDebit() {
        return this == WITHDRAWAL || this == TRANSFER;
    }

    public boolean isCredit() {
        return this == DEPOSIT || this == TRANSFER;
    }
}
```

Account.java

```
package com.intellibank.domain;

import jakarta.persistence.*;
import java.math.BigDecimal;
import java.util.UUID;
```

```java
@Entity
public class Account {

    @Id
    private UUID id;

    private String holderName;
    private BigDecimal balance;
    private boolean isFrozen;

    public Account() {}

    public Account(UUID id, String holderName, BigDecimal balance, boolean isFrozen)
{
        this.id = id;
        this.holderName = holderName;
        this.balance = balance;
        this.isFrozen = isFrozen;
    }

    public void credit(BigDecimal amount) {
        this.balance = this.balance.add(amount);
    }

    public void debit(BigDecimal amount) {
        if (this.balance.compareTo(amount) < 0)
            throw new IllegalArgumentException("Insufficient funds.");
        this.balance = this.balance.subtract(amount);
    }
}
```

AccountRepository.java

```java
package com.intellibank.repository;

import com.intellibank.domain.Account;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;
import java.util.UUID;

public interface AccountRepository extends JpaRepository<Account, UUID> {
    Optional<Account> findByHolderName(String holderName);
}
```

TransactionRequest.java

```java
package com.intellibank.service.dto;

import com.intellibank.domain.TransactionType;
import jakarta.validation.constraints.*;

import java.math.BigDecimal;
import java.util.UUID;

public record TransactionRequest(
    @NotNull UUID fromAccount,
    UUID toAccount,
    @NotNull @Positive BigDecimal amount,
    @NotNull TransactionType type
) {}
```

TransactionService.java

```java
package com.intellibank.service;

import com.intellibank.domain.Account;
import com.intellibank.domain.TransactionType;
import com.intellibank.repository.AccountRepository;
import com.intellibank.service.dto.TransactionRequest;
import jakarta.transaction.Transactional;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
public class TransactionService {

    private final AccountRepository accountRepository;

    @Transactional
    public void process(TransactionRequest request) {
        Account from = accountRepository.findById(request.fromAccount())
            .orElseThrow(() -> new IllegalArgumentException("Source account not found"));
        if (from.isFrozen()) throw new IllegalStateException("Source account is frozen.");

        if (request.type().isDebit()) {
            from.debit(request.amount());
        }

        if (request.type().isCredit() && request.toAccount() != null) {
            Account to = accountRepository.findById(request.toAccount())
                .orElseThrow(() -> new IllegalArgumentException("Destination account not
found"));
            to.credit(request.amount());
            accountRepository.save(to);
        }

        accountRepository.save(from);
    }
}
```

TransactionController.java

```
package com.intellibank.controller;

import com.intellibank.service.TransactionService;
import com.intellibank.service.dto.TransactionRequest;
import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/v1/transactions")
@RequiredArgsConstructor
public class TransactionController {

    private final TransactionService transactionService;

    @PostMapping
    public ResponseEntity<Void> execute(@RequestBody @Valid TransactionRequest
request) {
        transactionService.process(request);
        return ResponseEntity.ok().build();
    }
}
```

FraudDetectionService.java

```
package com.intellibank.service;

import org.springframework.stereotype.Service;

import java.math.BigDecimal;
import java.util.concurrent.CompletableFuture;

@Service
public class FraudDetectionService {

    public CompletableFuture<Boolean> isFraudulent(String accountHolder, BigDecimal
amount) {
        return CompletableFuture.supplyAsync(() -> {
            try {
                Thread.sleep(200); // simulate latency
            } catch (InterruptedException ignored) {}
            return amount.compareTo(BigDecimal.valueOf(10000)) > 0;
        });
    }
}
```

AuditSummaryCollector.java

```
package com.intellibank.util;

import java.math.BigDecimal;
import java.util.*;
import java.util.function.*;
import java.util.stream.Collector;

public class AuditSummaryCollector implements Collector<BigDecimal, BigDecimal[],
String> {

    @Override
    public Supplier<BigDecimal[]> supplier() {
        return () -> new BigDecimal[] { BigDecimal.ZERO, BigDecimal.ZERO };
    }
    @Override
    public BiConsumer<BigDecimal[], BigDecimal> accumulator() {
        return (acc, val) -> {
            acc[0] = acc[0].add(val); // total
            acc[1] = acc[1].max(val); // max
        };
    }

    @Override
    public BinaryOperator<BigDecimal[]> combiner() {
        return (acc1, acc2) -> new BigDecimal[] {
            acc1[0].add(acc2[0]),
            acc1[1].max(acc2[1])
        };
    }

    @Override
    public Function<BigDecimal[], String> finisher() {
        return acc -> "Total: " + acc[0] + ", Max: " + acc[1];
    }

    @Override
    public Set<Characteristics> characteristics() {
        return Set.of();
    }
}
```

TransactionServiceTest.java

```
package com.intellibank;

import com.intellibank.domain.Account;
import com.intellibank.domain.TransactionType;
import com.intellibank.repository.AccountRepository;
import com.intellibank.service.TransactionService;
import com.intellibank.service.dto.TransactionRequest;
import org.junit.jupiter.api.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

import java.math.BigDecimal;
import java.util.UUID;
```

```
@SpringBootTest
class TransactionServiceTest {

    @Autowired
    private AccountRepository repo;

    @Autowired
    private TransactionService service;

    private UUID fromId;
    private UUID toId;

    @BeforeEach
    void setup() {
        fromId = UUID.randomUUID();
        toId = UUID.randomUUID();
        repo.save(new Account(fromId, "Alice", BigDecimal.valueOf(5000), false));
        repo.save(new Account(toId, "Bob", BigDecimal.valueOf(2000), false));
    }

    @Test
    void testTransfer() {
        var request = new TransactionRequest(fromId, toId, BigDecimal.valueOf(1000),
TransactionType.TRANSFER);
        service.process(request);

        var from = repo.findById(fromId).get();
        var to = repo.findById(toId).get();

        Assertions.assertEquals(BigDecimal.valueOf(4000), from.getBalance());
        Assertions.assertEquals(BigDecimal.valueOf(3000), to.getBalance());
    }
}
```