@Entity

@Table(name = "TRADE")

1. ShareName

2. ShareQuantity

3. SharePrice

4. TraderId

5. BuyOrSell

6. Status

@RestController  
@RequestMapping("/api")  
public class TradeTransactionController {

@Cacheable(value = "trade-cache", key = "#tradeId")

**@GetMapping("/tradeTransactions/ {tradeId}")**public ResponseEntity<Trade> getById(@PathVariable("tradeId") String tradeId) {  
  
 Optional<Trade> tradeData = tradeService.findById(tradeId);  
 if (tradeData.isEmpty()) {  
 return new ResponseEntity<>(HttpStatus.NO\_CONTENT);  
 }  
 return new ResponseEntity<>(tradeData.get(), HttpStatus.OK);  
}

@CacheEvict(value = "trade-cache", key = "#tradeId")  
**@DeleteMapping("/tradeTransactions/{tradeId}")**  
public ResponseEntity<HttpStatus> deleteTrade (@PathVariable("tradeId") String tradeId) {  
 try {  
 tradeService.deleteById(tradeId);  
 return new ResponseEntity<>(HttpStatus.NO\_CONTENT);  
 } catch (Exception e) {  
 return new ResponseEntity<>(HttpStatus.INTERNAL\_SERVER\_ERROR);  
 }  
}

**@PostMapping("/tradeTransaction")**  
public ResponseEntity<Trade> createBankTransaction(@RequestBody Trade trade) {  
 try {  
 if(trade.getTradeId() == null || trade.getTradeId()) {

Trade.setTradeId(tradeService.generateTradeId);  
 Trade tradeNew = tradeService.save(trade);

shareExchangeService.notifyTradeInfo(tradeNew);  
 return new ResponseEntity<>( tradeNew, HttpStatus.CREATED);  
 } catch (Exception e) {  
 return new ResponseEntity<>(HttpStatus.INTERNAL\_SERVER\_ERROR);  
 }  
 }

}

@Service

Public class **TradeService** {

Private final TradeRepository tradeRepository;

**public Trade save(Trade trade)** {//Method to save a validated trade

boolean isValid = validateTrade(trade);

If(isValid) {

Trade.setStatus(“SUCCESS”);

} else {

trade.setStatus(“INVALID”);

return trade;

}

tradeRepository.save(trade);

handleBuyAction (trade);//for Buy action

handleSellAction (trade);//for Buy action

shareExchangeService.notify(trade);//Common step for all updates/inserts

}

**public Trade deleteById(Trade tradeId){}**//Method to remove trade

**public Trade findById(tradeId)** {} //Find trade by id

private Trade handleBuyAction(Trade trade) {

blockAmountInWallet(trade);//for Buy action

}

private Trade handleSellAction(Trade trade) {

blockQuantityInWallet(trade);//for sell action

}

private blockAmountInWallet(Trade trade) {}

private blockQuantityInWallet(Trade trade) {}

}

Design consideration

DB – Trade – RDBMS like Oracle storing the Trade master details

DB – Trade Transaction details – Can be stored in Cassandra as events with

Trade event,update notifications

Trade share details

Trade txns-qty,amount

Trade management service

Share Exchange Service

Monitor

API gateway

Cassandra/MongoDB

RDBMS