# LEOVEGAS
# JAVA ENGINEER
# CODING CHALLENGE

# WALLET MICROSERVICE

Build a simple wallet microservice running on the JVM that manages credit/debit transactions on behalf of players.

# DESCRIPTION

A monetary account holds the current balance for a player.
The balance can be modified by registering transactions on the account, either debit transactions (removing funds) or credit transactions (adding funds).
Create a REST API and an implementation that fulfils the requirements detailed below and honours the constraints.

# DESIRED FUNCTIONALITY

• Current balance per player

• Debit /Withdrawal per player A debit transaction will only succeed if there are sufficient funds on the account (balance - debit amount >= 0).
The caller will supply a transaction id that must be unique for all transactions. If the transaction id is not unique, the operation must fail.

• Credit per player. The caller will supply a transaction id that must be unique for all transactions. If the transaction id is not unique, the operation must fail.

• Transaction history per player

# NON-FUNCTIONAL REQUIREMENTS

• The solution need not persist data across restarts but it is a bonus if it does.

• If the solution uses a database, please use h2 or some other in-memory database. Make sure that your build/run script builds, installs and configures any such database.

• Do not use 3rd party software that entails us to install software on our machines. If 3rd party software is a necessity, create a docker image with a fully prepped environment.

## What we will look at:

- Design
- Clean code
- Testability
- Software craftsmanship

## What you shall think about:

- Concurrency
- Scalability
- Atomicityo
- Idempotency

# FINAL THOUGHTS

If you think some part of the exercise is unclear, don't worry.
Decide for yourself what would be a logical thing to do, and explain in your comments why you did what you did.

It is OK to reduce the scope of the assignment where necessary, but please provide a brief description of what would need to be changed in order for your service to be production-ready.
You are free to choose frameworks and libraries for the task yourself but please provide instructions on how to run your service and how the API looks.