Assessment: Java Engineer

You are starting a brand-new casino back-end system for a casino start-up. Normally you would need proper infrastructure in place, but the new CEO, Michael Scott, mentioned something about 'proof of concept' and this being a sort of 'assessment' and is only giving you very limited time. With this limited time there are only a select few crucial features this system should be capable of. A games provider, Dagacube, graciously offered to host a single slot game to get the casino started and have given some basic requirements on what the system should be capable of to host this slot game.

Dagacube's requirements:

They supplied an API specification with all the required functionality for casino slot games to function and be integrated with the casino.

Beyond the API specification, they warned that you should look into how the system would behave transaction wise if you receive several transactions within a short amount of time from each other for the same player, and that the balance updates correctly.

The Casino's requirements:

The system needs to make use of Java and Spring Boot.

Pam from customer support wants to be able to get the last ten wager/win transactions the player made on the slot game in order to better provide support. The Business Analyst, Dwight, took Dagacube’s API specification and added this requirement to it.

The CEO's requirements:

Michael approached you afterwards and mentioned that the system doesn't have to be all that fancy. He requested that you use a H2 database.
He also read up on some spring boot documents and mentioned to save time you only need to use one controller and service and promises he'll give you time afterwards to fix the inevitable tech debt.

He mentions his nephew, Toby, bragged that he can get a system like this running in less than 3 hours.

API Specification

Base Path: /casino

Get Balance

GET /player/{playerId}/balance

- Invalid playerId should be seen as a bad request (HTTP 400)

Request

Parameter	Location	Value	Description
playerId	path	integer	The player's id

Response

Parameter	Value	Description
playerId	integer	The player's id
balance	currency	The current balance of the player

Update Balance

POST /player/{playerid}/balance/update

- Invalid playerId should be seen as a bad request (HTTP 400)
- Negative amounts should be seen as a bad request (HTTP 400)
- Wager greater than current balance should be seen as a Teapot (HTTP 418)

Request

Parameter	Location	Value	Description
playerId	path	integer	The player's id
amount	body	currency (positive value)	The financial value of the transaction that is taking place
transactionType	body	static: WAGER WIN	States whether the update should be seen as a wager or a win

Response

Parameter	Value	Description
transactionId	big integer	The id of the transaction that took place
balance	currency	The player's current balance

Last 10 Transactions

POST /admin/player/transactions

- Invalid username should be seen as a bad request (HTTP 400)

Request

Parameter	Location	Value	Description
username	body	varchar, 50 length	The player's username for who the last ten transactions must be retrieved

Response

Parameter	Location	Value	Description
<Transactions array>	Root of body	-	The array in which the transactions must reside
transactionType	Transactions array (in transaction object)	static: WAGER WIN	States whether the transaction was of type wager or win
transactionId	Transactions array (in transaction object)	big integer	The id of the transaction that took place
amount	Transactions array (in transaction object)	currency	The financial value of the transaction