ATM Emulator

ATM is a part of our life activity, which helps us in day transactions and business. An automated teller machine (ATM) is a computerized telecommunications instrument that provides the clients of a financial institution with access to financial transactions in a public space without the need for a cashier, human clerk or bank teller.

At this time, the ATM provides the people good services especially the people can get money at any time. We need the ATM system because not all the bank branches are open all days of the week, and some of the customers may not in a situation, they can visit the bank every time, and they want to withdraw money or deposit money for emergency cases.

**ATM SPECIFICATIONS**

Overall, the core components of any ATM that will affect the interaction between ATM and its users are:

1. Cash-Dispenser: for dispensing cash.
2. Deposit-Slot: to deposit cash or checks from the users.
3. Printer: for printing the receipts.
4. Communication/Network Infrastructure: it is assumed that the ATM has a communication infrastructure to communicate with the bank upon any transaction or activity.

Implement and design the ATM system. It will act as a “second person” that a customer can “talk to”; the customer authenticates and makes choices that ATM replies back accordingly.

# Preferred development stack

1. Java as programming language,
2. Spring or Spring boot framework (version 2 and higher),
3. Any database (PostgreSQL, MongoDB, MySQL etc.)

# Mandatory functionality

1. User should be capable to set own preferred authentication method (example - PIN, fingerprint etc.),
2. Have a simplistic credit card validation mechanism when card number is provided ("Block" card after three unsuccessful attempts - should be present),
3. Based on the authentication method which user chose as "preferred" provide "session" initialization mechanism,
4. After authentication is successful initialized User should be capable to do standard list of operations with ATM: Cash deposit, Cash withdrawal, Check balance,
5. Limit number of entities involved in the infrastructure design (persistence layer) up to 5

# Optional

1. Having unit and Integration tests,
2. Having swagger configured,
3. Having method/class comments which follow to java document generation notations

# Important notes

1. Do not use JHipster or any similar code generation tools and/or frameworks,
2. Follow clean code rules and guidelines

# Expected Infrastructure Design

The overall project should consist from 2 microservices: "thin client" application (named as "atm-service", for example) and "bank-service".

Both of services are should support REST API development principles, no UI is necessary (Swagger can be a simple add-on for UI).

**For the "atm-service":**

1. There should be a session instantiation mechanism implemented, which can help to identify card number. Only **after** card is verified successfully, and corresponding response is sent, user can provide PIN for final login (like the real ATM does).
2. Bank should have a “preferred auth choice” connected to the card which will allow to provide as “PIN” method – 1. The actual PIN code, 2. Fingerprint “String”. This should be one of the points that should provide the interaction between “atm” and “bank” microservices.
3. "atm-service" should have no database connection, it should serve as a simple "thin client" for managing transactions,
4. Failover mechanism should be included for “atm-service” and “bank” MS-es interaction (Netflix Hystrix, Resilience4J or any other “Circuit Breaker” design pattern supported mean)

**For the "bank-service":**

1. All pointes from “Mandatory functionality” section expected to be implemented,
2. Provide a “secure interaction channel” between “atm-service” and “bank-service”

**On the “entire solution” level:**

1. Monitoring mechanism on all successful and unsuccessful transactions. Logging and logging configuration mechanism should be present, and stored in current microservice,
2. Exception handling mechanism is expected to be present,
3. Docker configuration is expected to be present.