Case Study: Banking Institution App

Objective:

Build an application that allows clients to manage their money in a succinct system that deals with multiple types of banking transactions.

Customers register to the application and receive detailed information on their account as well as tools to manage their assets.

In the Client view, the application must have atleast a home page with information you deem most important, a page for detailed account information, a quick access redirecting to the most popular client transactions and a page where the client can see their personal information and log out of their account.

System users:

- Banking App Admins
- Credit Card Companies
- Clients

Key High-Level Functional Features:

Some of the key high-level features of the system are:

1) The platform should allow the Banking App Admins to perform the following operations:
    a. Review account information of clients
    b. Add a client (in case a client asks to be manually added via phone or in person)
    c. Edit Client information (in case of change of address, etc.)
    d. Delete Client (in case the client closes their account)
    e. Edit/Set limits to an account's maximum transactional value (the amount allowed to use in a single day)
    f. Validate pictures of checks that are sent in to be deposited
    g. Add Institutions (for example: add SAAQ in the database as an institution that clients can add/select as recipients of a payment)
    h. Edit Institutions information
    i. Delete Institutions
    j. Add/Accept Credit Card Companies registration
    k. Edit Credit Card Companies accounts
    l. Delete Credit Card Companies accounts

Note: Other type of institutions include Universities, Insurance companies, City Council etc.

2) The system should allow Credit Card Companies to perform the following operations:
    a. Register their companies into a system where clients can browse the different products they offer.
    b. Request their account to be deleted/edited
    c. View client feedback on their different products

3) The app should allow the clients to perform the following operations:
    a. Transfer money between their accounts (a client might have 2 checking accounts and 1 credit card account with different amounts of money in them)
    b. Pay a bill (in reference to point 1g above)

   c. Send an e-transfer to another client (not necessarily of the same banking institution, but at least implement between same institution)
   d. Deposit Checks into a Checking account
   e. Edit personal information
   f. View Detailed information of all their accounts (including credit card scheduled payments and amounts due)
   g. Access a budgeting app (client inputs values and the budgeting app calculates a monthly budget plan – this inside-app logic is up to the dev)
4) To implement the features, the following specifications have been laid down to design the application:
   a. The design, layout, and navigation of the application should be different for each type of user logs in.
   b. Credit Card Company operations should be possible after Registration and Login.
   c. Admin operations should be possible using an admin route.
   d. Client operations should be possible after Registration and Login.
   e. The data provided by the user and credit card company must be validated.

All users must be able to log out/time out after inactivity

Key Non-Functional Features

1) App should be designed using Microservices architecture and implemented with polyglot persistence.
2) App should be responsive to display consistently across multiple device screens.
3) Containerize the entire application – the front-end and the backend (create Docker file, docker-compose.yml and get it executed through docker compose)
4) Use JWT token for Authentication
5) Usability, look and feel should be user friendly
6) Automation Testing must be included
7) CI should be implemented with Jenkins / Gitlab CI
8) Implementing Microservices patterns (API gateway, Config Server, Service Discovery-Eureka Server) and Messaging System (RabbitMQ)

Primary Service Modules

The domain specific microservices of the system are:

- Profile service
- Authentication service
- Account Management service
- CC service
- Search service
- Customer service
- Notification service

Other microservices for patterns and system management are:

- API Gateway service
- Discovery service
- Centralized configuration service
- Centralized logging service
- System monitoring service

Note: The system is assumed to be evolvable with addition and removal of services with new requirements