Flipkart UPI

**Background:**

Flipkart is aiming to enter the UPI (Unified Payments Interface) market to enhance its financial ecosystem and improve customer convenience. By integrating UPI, Flipkart can streamline payment processes, reduce dependency on third-party payment gateways, and leverage India's growing digital payment landscape. Additionally, this move aligns with its strategy to offer a seamless and integrated shopping experience, fostering customer loyalty and driving higher transaction volumes.

In order to achieve the above, Flipkart intends to integrate with a **Payment Service Provider (PSP) layer** so that it can achieve secure and efficient processing of UPI transactions, ensuring seamless fund transfers and real-time payment confirmations without excess regulatory overheads.

**Objective:**

Design and implement Flipkart UPI.

**Features:**

1. **User onboarding:** Users can be onboarded by calling an API and providing relevant details (name, phone number, active/deactive status).
2. **List of registered banks:** There should be a list of registered banks stored. Linking of bank accounts (next point) can be done only for registered banks. A bank should have the fields bank name and server status (up/down).
3. **Linking a Bank Account:** A user should be able to link multiple banks to an account. User Bank details include bank name, bank account number, bank balance (can be initialized a value).
4. Any one of the bank accounts has to be mandatorily linked to a phone number as a primary bank account.
5. **Making a payment:** A user should be able to make a payment to another user by providing username, phone number or bank account.
6. **Handling concurrency:** The transactions should be able to handle concurrency.
7. **Transaction history:** A user should be able to view all the transactions done at a bank account level and at an user level (cumulative of all bank accounts) [both amount paid and received].
8. A user should be able to search transactions based on other payee/payer users.

**Bonus Points:**

1. Search based on dates and transaction status.
2. **Handling pending payments transactions:** PSP layer can return a pending status for a transaction. However it is guaranteed that they’ll update the status within 120 seconds. Implement a retry mechanism while handling edge cases.

**Important Cases:**

1. A given phone number cannot be linked to more than one user account.
2. A given bank account cannot be linked to more than one user account.
3. A user account should be allowed to be marked as activated or deactivated.
4. Making a payment from one account to another should deduct the balance from the first account and add it to another account.
5. Payment can be made only from an active account.
6. Payment to an inactive account should throw relevant error.
7. Insufficient bank balance should be handled and throw proper exceptions.
8. If a bank server is down, any of the bank and payment operations should not be allowed.

**Other Notes:**

1. All the payment and bank related user transactions should be done by interacting with the PSP interface. [Bank list and status API should be written outside of this layer since it’s only for configuration and testing purposes].
2. Have a **driver class or API** to run the cases.
3. You can look for API references online.
4. Demoable code. Functionalities mentioned above.
5. **Functional completeness is a must.** Bonus questions are good to have.
6. Clean Interface design for the module.
7. Clean internal design and implementation of the library and the application.
8. Extensibility
9. Take care of Exception and Corner case handling.
10. Test cases covering various cases are good to have.
11. You are free to use **any language** of your choice.
12. **Use in memory data structures** for storing data.
13. UTs are good to have.