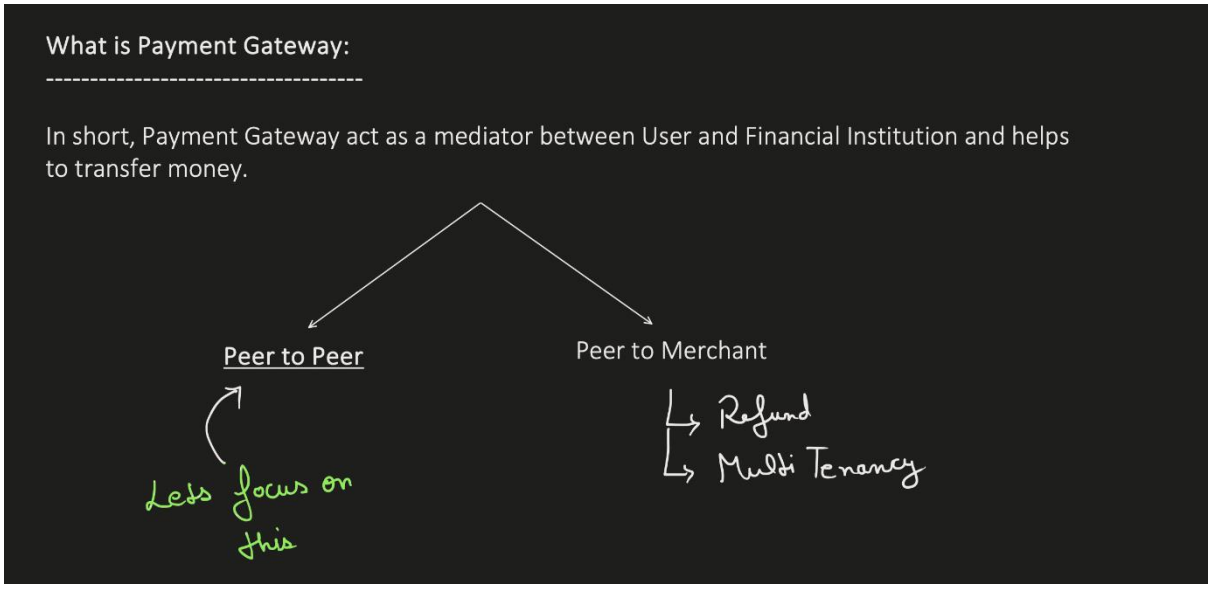
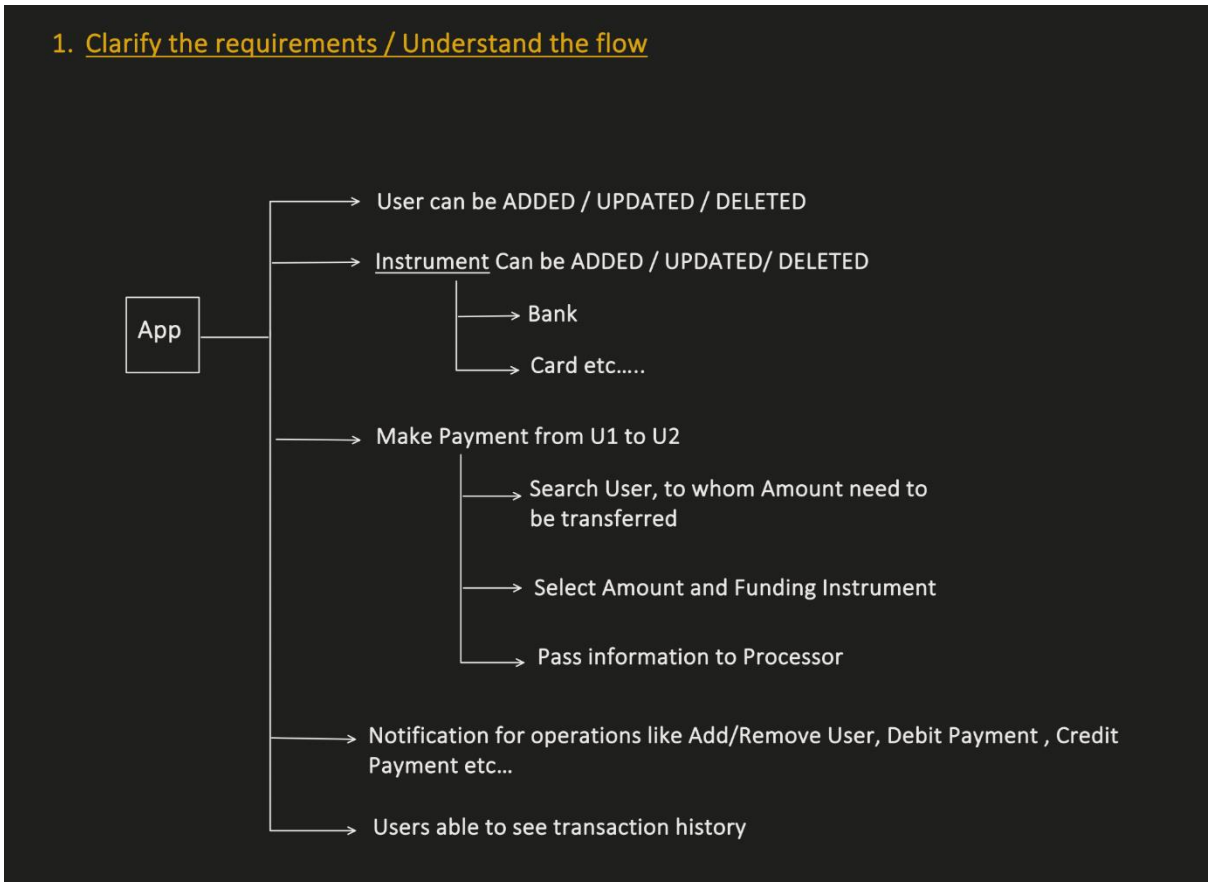
LLD: Design Payment gateway (Concept && Coding)

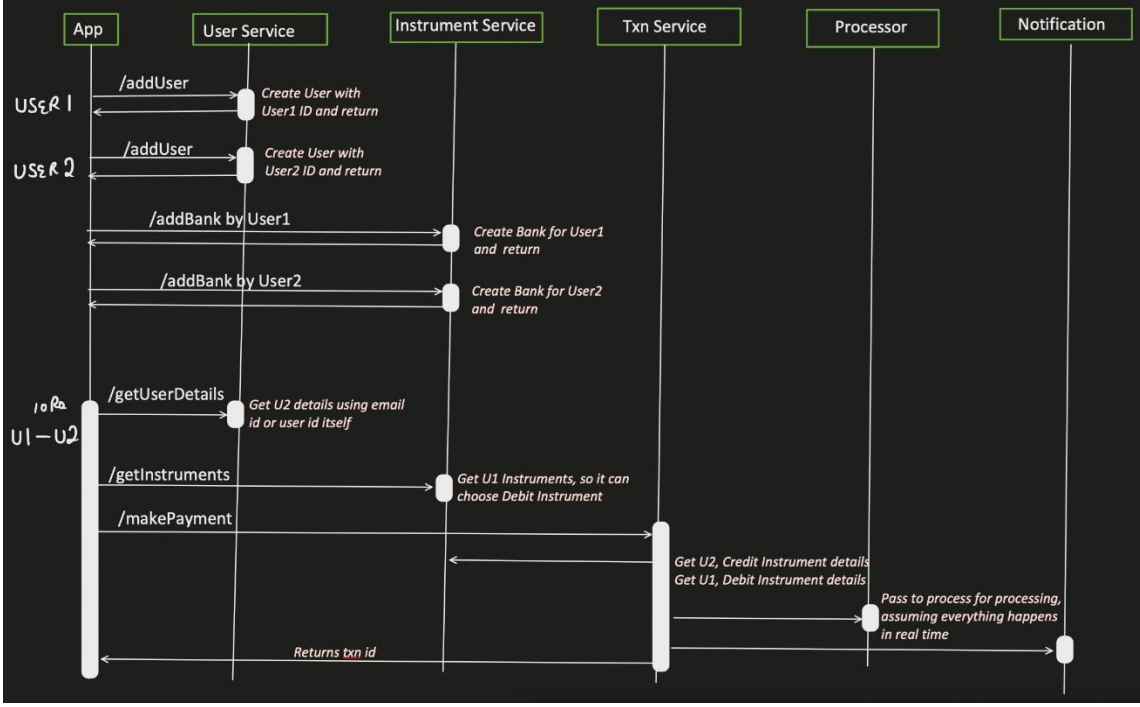




2. MAIN entity involved in the above flow

1. User
2. Instrument
    i. Bank,
    ii. Card etc...
3. Transaction
4. Transaction History
5. Notification
6. Processor

3. Sequencing Diagram of 1 happy flow:



Follow up question can come like:

1. Payment processing can take up to 3-5 days.

   We can make use of ASYC operation. Like
   - i) We will invoke Processor to validate,
     1. if U1 has sufficient money to fund this txn.
     2. If U2 instrument is valid.
   - ii) Then we can save the transaction in PENDING.
     1. We will invoke the process in ASYNC and complete the txn in PENDING state.
     2. Once Process Invoke our TXN Service and we will update the txn either SUCCESS or FAILURE.

```java
public class UserService {

    static List<User> usersList = new ArrayList<>();

    public UserDO addUser(UserDO userDO) {

        //some validations and create User obj

        User userObj = new User();
        userObj.setUserName(userDO.getName());
        userObj.setEmail(userDO.getMail());
        userObj.setUserID((int) new Random().nextInt( bound: 100-10)+10);
        usersList.add(userObj);
        return convertUserDOToUser(userObj);
    }

    public UserDO getUser(int userID) {
        for(User user : usersList){
            if(user.getUserID() == userID) {
                return convertUserDOToUser(user);
            }
        }
        return null;
    }

    private UserDO convertUserDOToUser(User userObj) {
        UserDO userDO = new UserDO();
        userDO.setName(userObj.getUserName());
        userDO.setMail(userObj.getEmail());
        userDO.setUserID(userObj.getUserID());
        return userDO;
    }
}
```

```java
public class UserDO {

    int userID;
    String name;
    String mail;

    public int getUserID() { return userID; }

    public void setUserID(int userID) { this.userID = userID; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getMail() { return mail; }

    public void setMail(String mail) { this.mail = mail; }
}
```

```java
public class UserController {

    UserService userService;

    public UserController() { userService = new UserService(); }

    public UserDO addUser(UserDO userDOObj) { return userService.addUser(userDOObj); }

    public UserDO getUser(int userID) { return userService.getUser(userID); }
}
```

```java
public class User {

    int userID;
    String userName;
    String email;

    public int getUserID() { return userID; }

    public void setUserID(int userID) { this.userID = userID; }

    public String getUserName() { return userName; }

    public void setUserName(String userName) { this.userName = userName; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }
}
```

```java
public class BankInstrument extends Instrument{

    String bankAccountNumber;
    String ifscCode;
}
```

```java
public class BankService extends InstrumentService {

    @Override
    public InstrumentDO addInstrument(InstrumentDO instrumentDO) {
        //bank specific logic here
        BankInstrument bankInstrument = new BankInstrument();
        bankInstrument.instrumentID = new Random().nextInt( bound: 100-10)+10;;
        bankInstrument.bankAccountNumber = instrumentDO.bankAccountNumber;
        bankInstrument.ifscCode = instrumentDO.ifsc;
        bankInstrument.instrumentType = InstrumentType.BANK;
        bankInstrument.userID = instrumentDO.userID;
        List<Instrument> userInstrumentsList = userVsInstruments.get(bankInstrument.userID);
        if(userInstrumentsList == null) {
            userInstrumentsList = new ArrayList<>();
            userVsInstruments.put(bankInstrument.userID, userInstrumentsList);
        }
        userInstrumentsList.add(bankInstrument);
        return mapBankInstrumentToInstrumentDO(bankInstrument);
    }

    public List<InstrumentDO> getInstrumentsByUserID(int userID) {
        List<Instrument> userInstruments = userVsInstruments.get(userID);
        List<InstrumentDO> userInstrumentsFetched = new ArrayList<>();
        for(Instrument instrument : userInstruments) {
            if(instrument.getInstrumentType() == InstrumentType.BANK)
            userInstrumentsFetched.add(mapBankInstrumentToInstrumentDO((BankInstrument) instrument));
        }
        return userInstrumentsFetched;
    }

    public InstrumentDO mapBankInstrumentToInstrumentDO(BankInstrument bankInstrument) {
        InstrumentDO instrumentDOObj = new InstrumentDO();
        instrumentDOObj.instrumentType = bankInstrument.instrumentType;
        instrumentDOObj.instrumentID = bankInstrument.instrumentID;
        instrumentDOObj.bankAccountNumber = bankInstrument.bankAccountNumber;
        instrumentDOObj.ifsc = bankInstrument.ifscCode;
        instrumentDOObj.userID = bankInstrument.userID;
        return instrumentDOObj;
    }
}
```

```java
public class CardInstrument extends Instrument{

    String cardNumber;
    String cvvNumber;
}
```

```java
public class CardService extends InstrumentService{

    @Override
    public InstrumentDO addInstrument(InstrumentDO instrumentDO) {
        //card specific logic here
        CardInstrument cardInstrument = new CardInstrument();
        cardInstrument.instrumentID = new Random().nextInt( bound: 100-10)+10;
        cardInstrument.cardNumber = instrumentDO.cardNumber;
        cardInstrument.cvvNumber = instrumentDO.cvv;
        cardInstrument.instrumentType = InstrumentType.CARD;
        cardInstrument.userID = instrumentDO.userID;
        List<Instrument> userInstrumentsList = userVsInstruments.get(cardInstrument.userID);
        if(userInstrumentsList == null) {
            userInstrumentsList = new ArrayList<>();
            userVsInstruments.put(cardInstrument.userID, userInstrumentsList);
        }
        userInstrumentsList.add(cardInstrument);
        return mapBankInstrumentToInstrumentDO((CardInstrument) cardInstrument);
    }

    public List<InstrumentDO> getInstrumentsByUserID(int userID) {
        List<Instrument> userInstruments = userVsInstruments.get(userID);
        List<InstrumentDO> userInstrumentsFetched = new ArrayList<>();
        for(Instrument instrument : userInstruments) {
            if(instrument.getInstrumentType() == InstrumentType.CARD)
                userInstrumentsFetched.add(mapBankInstrumentToInstrumentDO((CardInstrument) instrument));
        }
        return userInstrumentsFetched;
    }

    public InstrumentDO mapBankInstrumentToInstrumentDO(CardInstrument cardInstrument) {
        InstrumentDO instrumentDOObj = new InstrumentDO();
        instrumentDOObj.instrumentType = cardInstrument.instrumentType;
        instrumentDOObj.instrumentID = cardInstrument.instrumentID;
        instrumentDOObj.cardNumber = cardInstrument.cardNumber;
        instrumentDOObj.cvv = cardInstrument.cvvNumber;
        instrumentDOObj.userID = cardInstrument.userID;
        return instrumentDOObj;
    }
}
```

```java
public abstract class Instrument {

    int instrumentID;
    int userID;
    InstrumentType instrumentType;

    public int getInstrumentID() { return instrumentID; }

    public void setInstrumentID(int instrumentID) { this.instrumentID = instrumentID; }

    public int getUserID() { return userID; }

    public void setUserID(int userID) { this.userID = userID; }

    public InstrumentType getInstrumentType() { return instrumentType; }

    public void setInstrumentType(InstrumentType instrumentType) { this.instrumentType = instrumentType; }
}
```

```java
public class InstrumentDO {

    int instrumentID;
    String ifsc;
    String cvv;
    String bankAccountNumber;
    String cardNumber;
    InstrumentType instrumentType;
    int userID;
```

```java
public class InstrumentDO {

    int instrumentID;
    String ifsc;
    String cvv;
    String bankAccountNumber;
    String cardNumber;
    InstrumentType instrumentType;
    int userID;

    public int getInstrumentID() { return instrumentID; }

    public void setInstrumentID(int instrumentID) { this.instrumentID = instrumentID; }

    public String getIfsc() { return ifsc; }

    public void setIfsc(String ifsc) { this.ifsc = ifsc; }

    public String getCvv() { return cvv; }

    public void setCvv(String cvv) { this.cvv = cvv; }

    public String getBankAccountNumber() { return bankAccountNumber; }

    public void setBankAccountNumber(String bankAccountNumber) { this.bankAccountNumber = bankAccountNumber; }

    public String getCardNumber() { return cardNumber; }

    public void setCardNumber(String cardNumber) { this.cardNumber = cardNumber; }

    public InstrumentType getInstrumentType() { return instrumentType; }

    public void setInstrumentType(InstrumentType instrumentType) { this.instrumentType = instrumentType; }

    public int getUserID() { return userID; }

    public void setUserID(int userID) { this.userID = userID; }
}
```

```java
public abstract class InstrumentService {
    static Map<Integer, List<Instrument>> userVsInstruments = new HashMap<>();

    public abstract InstrumentDO addInstrument(InstrumentDO instrumentDO);

    public abstract List<InstrumentDO> getInstrumentsByUserID(int userID);

}
```

```java
public enum InstrumentType {

    BANK,
    CARD;
}
```

```java
public class TransactionController {

    TransactionService txnService;

    public TransactionController() { txnService = new TransactionService(); }

    public TransactionDO makePayment(TransactionDO txnDO) {
       return txnService.makePayment(txnDO);
    }

    public List<Transaction> getTransactionHistory(int userID) { return txnService.getTransactionHistory(userID); }
}
```

```java
public class TransactionService {

    public static Map<Integer, List<Transaction>> userVsTransactionsList = new HashMap<>();
    InstrumentController instrumentController;
    Processor processor;

    public TransactionService(){
        instrumentController = new InstrumentController();
        processor = new Processor();
    }

    public  List<Transaction> getTransactionHistory(int userID) { return userVsTransactionsList.get(userID); }
        public TransactionDO makePayment(TransactionDO txnDO) {
        InstrumentDO senderInstrumentDO = instrumentController.getInstrumentByID(txnDO.getSenderId(), txnDO.getDebitInstrumentId());
        InstrumentDO receiverInstrumentDO = instrumentController.getInstrumentByID(txnDO.getReceiverId(), txnDO.getCreditInstrumentId());
        processor.processPayment(senderInstrumentDO, receiverInstrumentDO);
        Transaction txn = new Transaction();
        txn.setAmount(txnDO.getAmount());
        txn.setTxnID(new Random().nextInt( bound: 100-10)+10);
        txn.setSenderId(txnDO.getSenderId());
        txn.setReceiverId(txnDO.getReceiverId());
        txn.setDebitInstrumentId(txnDO.getDebitInstrumentId());
        txn.setCreditInstrumentId(txnDO.getCreditInstrumentId());
        txn.setStatus(TransactionStatus.SUCCESS);
        List<Transaction> senderTxnsList = userVsTransactionsList.get(txn.getSenderId());
        if(senderTxnsList == null) {
            senderTxnsList = new ArrayList<>();
            userVsTransactionsList.put(txn.getSenderId(), senderTxnsList);
        }
        senderTxnsList.add(txn);
        List<Transaction> receiverTxnLists = userVsTransactionsList.get(txn.getReceiverId());
        if(receiverTxnLists == null) {
            receiverTxnLists = new ArrayList<>();
            userVsTransactionsList.put(txn.getReceiverId(), receiverTxnLists);
        }
        receiverTxnLists.add(txn);
        txnDO.setTxnID(txn.getTxnID());
        txnDO.setStatus(txn.getStatus());
        return txnDO;
    }
```

```java
public class PaymentGateway {

    public static void main(String args[]) {

        InstrumentController instrumentController = new InstrumentController();
        UserController userController = new UserController();
        TransactionController transactionController = new TransactionController();

        //1. add USER1
        UserDO user1 = new UserDO();
        user1.setName("Sj");
        user1.setMail("sj@conceptandcoding.com");
        UserDO user1Details = userController.addUser(user1);

        //1. add USER2
        UserDO user2 = new UserDO();
        user2.setName("Pj");
        user2.setMail("Pj@conceptandcoding.com");
        UserDO user2Details =userController.addUser(user2);

        //add bank to User1
        InstrumentDO bankInstrumentDO = new InstrumentDO();
        bankInstrumentDO.setBankAccountNumber("234324234324324");
        bankInstrumentDO.setInstrumentType(InstrumentType.BANK);
        bankInstrumentDO.setUserID(user1Details.getUserID());
        bankInstrumentDO.setIfsc("ER3223E");
        InstrumentDO user1BankInstrument = instrumentController.addInstrument(bankInstrumentDO);
        System.out.println("Bank Instrument created for User1: " + user1BankInstrument.getInstrumentID());

        //add Card to User2
        InstrumentDO cardInstrumentDO = new InstrumentDO();
        cardInstrumentDO.setCardNumber("1230099");
        cardInstrumentDO.setInstrumentType(InstrumentType.CARD);
        cardInstrumentDO.setCvv("0000");
        cardInstrumentDO.setUserID(user2Details.getUserID());
        InstrumentDO user2CardInstrument = instrumentController.addInstrument(cardInstrumentDO);
        System.out.println("Card Instrument created for User2: " + user2CardInstrument.getInstrumentID());
```

```
        //make payment
        TransactionDO transactionDO = new TransactionDO();
        transactionDO.setAmount(10);
        transactionDO.setSenderId(user1Details.getUserID());
        transactionDO.setReceiverId(user2Details.getUserID());
        transactionDO.setDebitInstrumentId(user1BankInstrument.getInstrumentID());
        transactionDO.setCreditInstrumentId(user2CardInstrument.getInstrumentID());
        transactionController.makePayment(transactionDO);

        //get All instruments of USER1
        List<InstrumentDO> user1Instruments = instrumentController.getAllInstruments(user1Details.getUserID());
        for(InstrumentDO instrumentDO : user1Instruments) {
            System.out.println("User1 InstID: " + instrumentDO.getInstrumentID() +
                    ": UserID: " +instrumentDO.getUserID() +
                    ": InstrumentType: " +instrumentDO.getInstrumentType().name());
        }

        //get All instruments of user2
        List<InstrumentDO> user2Instruments = instrumentController.getAllInstruments(user2Details.getUserID());
        for(InstrumentDO instrumentDO : user2Instruments) {
            System.out.println("User2 InstID: " + instrumentDO.getInstrumentID() +
                    ": UserID: " +instrumentDO.getUserID() +
                    ": InstrumentType: " +instrumentDO.getInstrumentType().name());
        }

        //get All transaction history
        List<Transaction> user1TransactionList = transactionController.getTransactionHistory(user1Details.getUserID());
        for(Transaction txn : user1TransactionList) {...}
    }
}
```