The challenge

Build a bank API using Python and either FastAPI or Django. The API should include full CRUD functionality for bank accounts and transactions and should be able to handle the following operations:

API Routes to be implemented

1. Retrieve an existing bank account
2. Update an existing bank account
3. Delete an existing bank account
4. Create a new transaction
5. Retrieve an existing transaction
6. Update an existing transaction
7. Delete an existing transaction

Account Model:

1. account_number (string) - unique identifier for the bank account
2. balance (float) - current balance in the account
3. customer_name (string) - name of the customer
4. account_type (string) - type of account (e.g. savings, checking)

Transaction Model:

1. transaction_id (string) - unique identifier for the transaction
2. account_number (string) - account number associated with the transaction
3. amount (float) - amount involved in the transaction
4. transaction_type (string) - type of transaction (e.g. deposit, withdrawal)
5. timestamp (datetime) - timestamp of the transaction

Constraints

Account Model:

- The account number should be unique across all accounts.
- The balance should be updated automatically based on the transactions associated with the account.
- Consider including additional fields like date of account creation, date of last update.
- Consider adding validation for initial deposit
- Consider adding constraints for account type and account number

Transactions:

- The transaction id should be unique across all transactions.
- The account number should be a valid account number that exists in the bank accounts.
- The amount should be a positive number.
- The transaction type should be one of a predefined set of options (e.g. deposit, withdrawal)
- The timestamp should be in a valid format and should be set automatically when the transaction is created.
- The account balance should be updated automatically based on the transaction type and amount.
- Consider adding fields like transaction description and transaction status
- Consider adding constraints for transaction type and transaction id

Example creating a new transaction with the following information:

```
{
    "transaction_id": "abcdefghijk",
    "account_number": "1234567890",
    "amount": 500.00,
    "transaction_type": "deposit",
    "description": "Initial deposit",
    "status": "success"
}
```

- transaction_id: a unique identifier for the transaction
- account_number: the account number associated with the transaction
- amount: the amount involved in the transaction, which is 500.00
- transaction_type: the type of transaction, which is a deposit
- description: additional information about the transaction, which is "Initial deposit"
- status: the status of the transaction, which is "success"

Also, you should consider the constraints that are related to the transaction such as the transaction type should be "deposit" or "withdrawal" only and the account_number should exist in the bank accounts table.

Example of how to decrease the balance of a bank account in response to a withdrawal transaction:

- When a withdrawal transaction is created, the API should first retrieve the current balance of the associated bank account using the account number from the transaction.
- The API should then subtract the amount of the withdrawal from the current balance.
- The API should then update the balance of the bank account in the database with the new balance.
- Here is an example of the JSON input for a withdrawal transaction:

```
{
    "transaction_id": "abcdefghijk",
    "account_number": "1234567890",
    "amount": 500.00,
    "transaction_type": "withdrawal",
    "description": "ATM withdraw",
    "status": "success"
}
```

When this transaction is created, the API should retrieve the bank account with the account number "1234567890" and decrease the balance by 500.00.

This can be implemented using the following code snippet:

```
    account =
BankAccount.objects.get(account_number=transaction["account_number"])
    account.balance -= transaction["amount"]
    account.save()
```

Please consider concurrency scenarios when updating the balance of a bank account:

Relationships between Bank Account and Transactions:

- Consider the relationship between the two models. If it is one-to-many, meaning one bank account can have multiple transactions, you should consider adding a foreign key from the transactions model to the bank accounts model.

- Consider creating a separate endpoint for retrieving all transactions associated with a specific bank account.

- Consider creating a separate endpoint for updating the balance of an account after a transaction is made

Testing:

- Consider using a testing framework like Pytest or Unittest to test the API.
- Consider creating test cases for all relevant above-mentioned operations