Entities

Promo_codes
- Id
- Title
- Desc
- Benefit (cents) int
- Quantity int
- Code (value of the code, to be shared with users)
- Start_date
- End_date
- Is_active
- Freq_of_use int (max no uses per user)

User
- Id (PK)
- Username (unique)
- Email (unique)
- Timestamp

Promo_history: (relationship table)
- Id
- User_id (fk)
- Promo_code_id (fk)
- Transaction_id (fk)
- Freq-of-user for a user id (calculated from this table)

Transactions
- Id
- Created_ts
- Promo_code_id (nullable)
- Amount (total) (store amount in cents) it must be int
- Billed_amount (charged after applying promo code)
- Validation (billed_amount + promo benefit == amount)
- User_id (fk) (customer, client)
- Currency
- State (enum) (created, pending, processed, or failed)
- Payment_method (creditcard, cash, ewallet)

Extras (if you have more time)
- Recipient ID (ex. merchant)
-

Example use case:

If we have a ride for (amount) 50 LE. And you have a promo code for 10LE. Ride charge should be (billed_amount) 40LE

Assumptions:

1. User can only apply 1 promo code per transaction
2. The endpoint will be used by a frontend client (web or mobile app)

Endpoints

promocodes/

METHOD POST
Request BODY JSON {user_email, promocode (promo_codes.code), transaction_id}

Response
SUCCESS

- Promocode, billed_amount, transaction_id, remaining_no_of_uses (promo_code.freq - calculated freq from promo_history)

ERRORS

- Promo didn't start
- Promo expired
- Exceed freq_of_use
- Promo code inactive
- Promo code not found
- Invalid or missing email, code, transaction_id

Steps

1. Create Django models (representing ER diagram)
2. Install DRF
3. Create APIView for the
4. Create Serializers for request and response for each endpoint
5. If necessary create validators
6. You need to decide where will business logic

```
Class Promocode(django.models.Model):
    # … attr

    def redeem_promocode(promocode_code, txn_id, user_email):
        If now() < start_date:
          ….
       If ..



Class User:
   Def no_of_uses(self, promocod_id):
        # query promo_history for self.id and promocode_id and return
the count


Diab = user.no_of_users(12346567)
```

General advice

0. Use use pytest, virtualenv for deps management
1. Draw quick flow chart for the business logic to have a good idea about the general cases and the corner ones as well
1. Think about the best way to break down the business logic
2. Make sure relationships between entities is covered in models and called out in your assumptions
3. Write unit tests only for critical model functions, then write (if you have time) integration tests for the API
4. For documentation, look up pydoc strings (in code documentation)
5. Make sure to write a clear, well-organized readme
    a. SHould include brief description for the project
    b. Assumptions you made
    c. THings you would've made if you had more time
    d. Installation instructions
    e. Use (how to use the API, how to run the tests)
6. Obviously, share the code via git
7. Make sure the installation and the use steps actually work (try reinstalling and running the app from scratch once you're done to make sure the env is reproducible)

https://github.com/Sheshtawy/hawkeye