Bright Money: Assessment for Backend [SDE - 1] .

Dear Candidate,

Thank you for your interest in Bright. As part of our hiring process, you shall be required to submit the assessment below. Please note that the deadline to submit your responses is in the next **48 Hours.**

For any questions, please feel free to reach out.

Problem Statement

You will be designing a Loan Management system which will help a Loan service company to effectively lend loans to users.

For the given problem, you will have to make a Django application which will have the stated functionalities and features.

Relevant models, views, APIs, flows and administrative aggregation have to be designed and implemented.

You will have to logically persist the data required in flows and fetch it via Django ORM where needed.

You will be judged on several factors including but not limited to how you are creating models, managing authentication, writing APIs and serializers, handling flows, OOP principles, code clarity and how to run your project.

Provide a name to the project. Strict plagiarism checks will be imposed and any copy found will result in disqualification.

A .csv file will be provided with the application which contains the details of the user's savings account transactions.

Fields: -

AADHAR ID: unique user id

- Date: date of transaction
- Amount: amount of transaction
- Transaction_type: DEBIT or CREDIT. CREDIT is the amount added in the account and vice versa Note that this will contain some sample data. You can add more data to check your flows.

Functionalities Required: -

POST API View to register a User's (availing the loan) details

- Endpoint: /api/register-user/
- Codes: 200,400
- View: Create a User with following details
- Name
- Email
- Annual Income
- An async celery task must be triggered via this API to calculate the credit score of the user.
- The logic for calculation of credit score is defined as
- The credit score calculation must be done using the transactions from the csv file.
- Net credit score minimum and maximum range is 300 and 900. Credit score can't be

Fractional.

- For total account balance (computed as sum(CREDIT - DEBIT) amounts across all transactions for the given User) -> If value >= 10,00,000, then Credit Score is maximum possible value (900), Else if value <= 1,00,0000 then Credit Score value is minimum (300).
- For any intermediate account balance value, Credit score changes by 10 points for every Rs. 15,000 change in account balance.
- Example for an account balance of Rs.5,68,450, credit score is 670.

Request Fields

- Aadhar ID: Unique User Identifier already generated and the same is given in csv.
- name
- email_id
- annual_income

Response Fields

- Error: None if success. Error string in case ingestion was not successful stating apt reason
- unique_user_id (UUID value) generated for the given User

POST API View to apply & initiate a User Loan

- Endpoint: /api/apply-loan/
- Codes: 200,400
- View: Create a User Loan Application complete with the requisite details subject to the below conditions. The user registration should be done before applying for a loan.
- Following Type of Loans are supported (‘Car’, ‘Home’, ‘Education’, ‘Personal’)
- Application will only be created if the User credit_score >= 450. No loans will be disbursed if the credit score is not found in the system.
- User Income must be >= Rs. 1,50,000 annually.
- Loan Amount requested must be within bounds dependent upon the Type of Loan applied for:
- Car: Rs.7,50,000
- Home: Rs.85,00,000
- Educational: Rs.50,00,000
- Personal: Rs.10,00,000

EMIs calculation

- Interest is incurred starting from next day of disbursal
- All due amounts should be paid back within tenure. All EMI amounts should be the same except the last which can be less than the other EMIs. Note
- last EMI amount settles all pending dues.
- EMIs consists of (formula to be deduced by you)
- Part of Principal amount due

- Interest on that principal amount for that month
- EMI amount must be at-most 60% of the monthly income of the User
- Interest rate should be >=14%
- Total interest earned should be >10000
- All EMIs will be due on 1st of every month starting from the following month of disbursement -

Request Fields

- unique_user_id: Unique User Identifier (UUID)
- loan_type: Loan type as supported above
- loan_amount: Amount of Loan in rupees
- interest_rate: Rate of interest in percentage
- term_period: time period of repayment in months
- disbursement_date: Date of disbursal

Response Fields

- Error: None if success. Error string in case ingestion was not successful stating apt reason.
- Loan_id: Unique identifier to identify the loan initiated.
- Due_dates: List containing objects of EMI dates and amount. Each object contains:
- Date: EMI date
- Amount_due: Amount due on that date

POST API to register payment made

- Endpoint: /api/make-payment/
- Codes: 200,400
- View: Ingest data with the payment made towards an EMI.
- Reject payment if
- payment is already made for that date
- Previous EMIs are due
- EMI amount should be recalculated if the Amount being paid by the User is less/more than the due installment amount.
- Request fields
- Loan_id: Unique identifier to identify the loan

- Amount (depends upon User)
- Response
- Error: None if no error. Error String in case of failure

GET API to get statement and future dues

- Endpoint: /api/get-statement/
- Codes: 200,400

- View:

- Return error if loan does not exists of closed
- For a valid loan
- Statement of amount paid with details of the following. For sake of simplicity assume all transactions (interest calculation and payment) were due/ done on 1st of every month
- Principal due
- Interest on Principal due for that month
- Repayment of principal
- Statement of all upcoming EMIs

Request fields

- Loan_id: Loan for which details are to be fetched

Response fields

- Error: None if no error. Error String in case of failure
- Past_transactions: Empty list for no past transactions. For valid transactions, each object contains:
  - Date
  - Principal
  - Interest
  - Amount_paid
  - Upcoming_transactions: List containing objects of EMI dates and amount. Each object contains:
  - Date: EMI date
  - Amount_due: Amount due on that date.

All The Best