Documentation

Aspire-Mini API

K Nithiya Soundari

Github link: https://github.com/Nithiya99/loan-api

Overview

The Aspire-Mini API is an app that has all the basic features of loans. This app has been developed with the **Laravel 9** framework and **MySQL** as the backend database. Database design, database schema and design choices are discussed in this documentation.

Design Choices

For this project, I have assumed the users who are available under the user table are the authenticated customers or admin(s). The user is determined as 'Client' (customer) or 'Admin' depending on the 'user_type'. I have assumed so as I only had to develop the backend APIs for this project. In normal scenario, I will do the authentication through session token.

Here, I will discuss some of the design choices I made for each developed API. The routes, parameters required, sample calls and responses are all discussed in detail in github (https://github.com/Nithiya99/loan-api#documentation)

1. Customer Submit Loan Request
   The user needs to send the **loan amount** [loan_amt], **loan terms** [loan_terms] and **customer ID** [cust_id].
   a. Loan amount must be greater than $0 and has a request limit amount of ten million dollars.
   b. Loan terms must be greater than 0.

c. It is checked if the user is genuine (registered & authenticated) and if the user is of type Client (customer).

If all the parameters is valid, the loan details are saved in the loans table and the per loan term details are generated and saved in the loan_payments table.

2. Admin Approve Loan Request

   The admin sends his user ID and loan ID.

   a. Fist it is checked if the admin exists in the users table
   b. It is checked if a loan with the given ID exists and whether the status of the loan is PENDING.
   c. If is is PENDING, the status is updated to APPROVED and the admin's ID and the time at which the admin approved the loan is recorded.

3. Customer Views Loans that Belong to him

   The customer can view the loans that he has under him through this route.

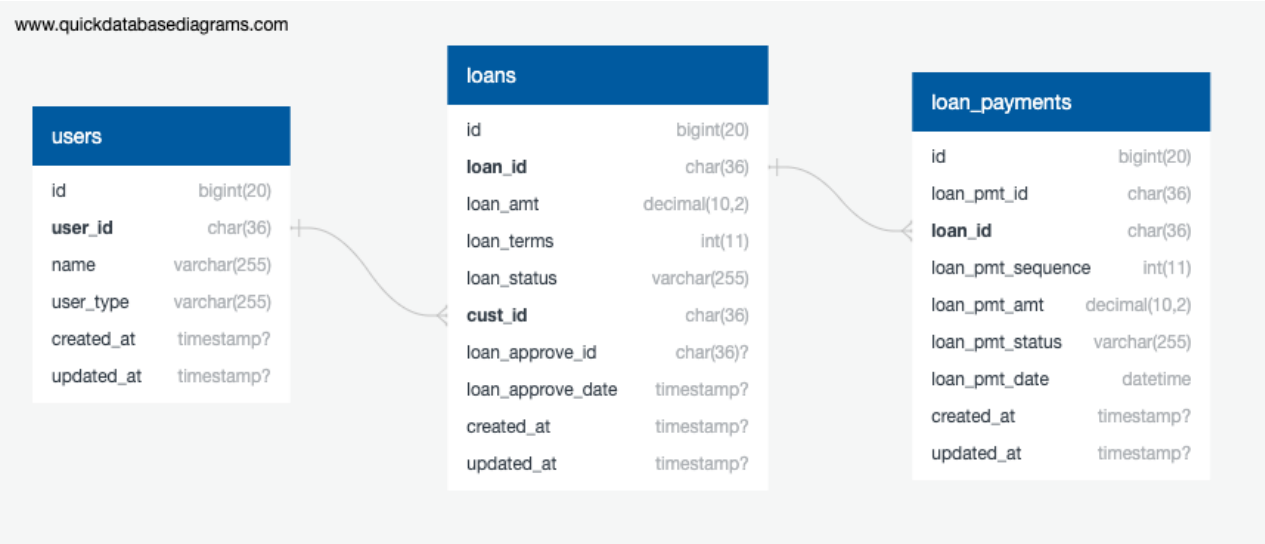
4. Customer Repay Loan

   User sends the repayment amount and the loan ID for the loan he wishes to pay for.

   a. The repayment amount he sends should me greater than or equal to the term repayment amount.
   b. If the money is greater than the term's repayment, the remaining money will be used to partially pay of the upcoming term's payment.
   c. If all the terms for the loan are paid of and there is some money remaining, the remaining money is returned back to the customer.

Database Design

Below diagram depicts the database schema.



Database Name: loanDB

The database consist of 3 tables:

1. **users**

   In this coding challenge, it is assumed that the users who exist in the table are authenticated. In real world scenario, the user will be checked if he is an authenticated user through his session token.

Column Name	Data Type	Additional Info
id	Int - autoincrement	
user_id	String - unique - uuid	Primary Key
name	String	
user_type	String	‘Client’ or ‘Admin’

created_at	timestamp	
updated_at	timestamp	

2. loans

   When a customer submits a loan request, if the loan is successfully created, the loan record will be stored in this table.

Column Name	Data Type	Additional Info
id	Int - autoincrement	
loan_id	String - unique - uuid	PK
loan_amt	decimal	
loan_terms	int	
loan_status	String	'PENDING' when loan request submitted
cust_id	String - uuid	FK->users->user_id
loan_approve_id	String - nullable	FK->users->user_id
loan_approve_date	timestamp	
created_at	timestamp	
updated_at	timestamp	

3. loan_payaments

   Each term loan payment information will be saved in this table. Till the loan term due is paid, the status of that term will be PENDING.

Column Name	Data Type	Additional Info
id	Int - autoincrement	

loan_pmt_id	String - unique - uuid	PK
loan_id	String - uuid	FK->loans->loan_id
loan_pmt_sequence	int	‘PENDING’ till loan is paid. ‘PAID’ once loan is paid.
loan_pmt_amt	decimal	
loan_pmt_status	string	
loan_pmt_date	timestamp	
created_at	timestamp	
updated_at	timestamp	

Conclusion

This app has loan API routes for all the four functions mentioned above. It was a good experience developing this app in Laravel. I was able to learn and comprehend the advantages of using Laravel and why Laravel is such a popular language in the industry. Through this app development, I realized that Laravel is such a powerful framework that makes backend development easier and more clean. I have only grazed the surface of Laravel through this challenge, and I am excited to learn more and go deeper into all the functionalities that Laravel offers.