Document Technical Dating Apps

Author : Moh. Radyatama Suryana

Email : mohradyatama24@gmail.com

1. Url github repository: https://github.com/radyatamaa/dating-apps-api
2. List Of Functionality :
   a. User Register

      Endpoint: POST /api/v1/user/register

      Description: This endpoint allows new users to sign up for the app. The user provides a name, age, bio, photo, email, and password. The password is hashed before being stored in the database for security.

   b. User Login

      Endpoint: POST /api/v1/user/login

      Description: This endpoint allows users to log in by providing their email and password. If the credentials are valid, a JWT token is returned for authenticated access to other endpoints.

   c. Purchasing Premium

      Endpoint: POST /api/v1/user/purchase-premium

      Description: This endpoint allows users to purchase premium packages that unlock additional features (e.g., no swipe quota, verified label). The purchase is processed and the user's premium status is updated, needed ` Authorization: Bearer <JWT_TOKEN>` for the update status premium.

   d. Update Location of Current User

      Endpoint: PUT /api/v1/profile/location

      Description: This endpoint allows users to update their current location. This location data can be used to show nearby profiles, needed ` Authorization: Bearer <JWT_TOKEN>` for the update current location.

   e. Profile Viewing

      Endpoint: GET /api/v1/profile

      Description: This endpoint allows users to view a list of profiles with pagination and longitude latitude request for nearby profile, they can swipe on. It enforces the daily swipe limit (10 profiles per day for non-premium users) and ensures the same profile does not appear twice in the same day (if pass the profile will doesn't appear at the same day , if liked will doesn't appear again), needed ` Authorization: Bearer <JWT_TOKEN>` for the profile viewing.

   f. Swiping

      Endpoint: POST /api/v1/swipe/profile

      Description: This endpoint allows users to swipe left (pass) or right (like) on a profile. It records the swipe action and enforces the daily limit for non-premium users (like + pass each profile max 10 profile per day), needed ` Authorization: Bearer <JWT_TOKEN>` for the profile viewing.

3. List of the tech stacks has been used :
    a. Golang : Go is chosen for its high performance and efficiency as a statically typed, compiled language. It excels in handling concurrent operations, making it suitable for applications requiring real-time interactions, like a dating app. Go's built-in support for concurrency using goroutines and channels simplifies managing multiple user interactions simultaneously. Additionally, Go's simple syntax enhances code readability and maintainability, reducing complexity and easing the onboarding process for new developers.

    b. Beego Framework : Beego is organizes the codebase in a structured manner, facilitating easier management and scalability. It comes with built-in features such as ORM, session management, and logging, which expedite development and reduce the reliance on additional third-party libraries. Additionally, Beego benefits from a growing community and comprehensive documentation, providing valuable support and best practices, and also least issues instead of others framework like gin/echo base on github.

    c. Clean Architecture : Clean Architecture is adopted to ensure the separation of concerns and maintainability of the codebase. It promotes a clear distinction between the business logic and the infrastructure, making the system more modular and testable. By organizing the code into layers, Clean Architecture facilitates easier updates and scalability, as changes in one layer have minimal impact on others. This approach also enhances the readability and understandability of the code, allowing developers to quickly grasp the system's structure and functionality.

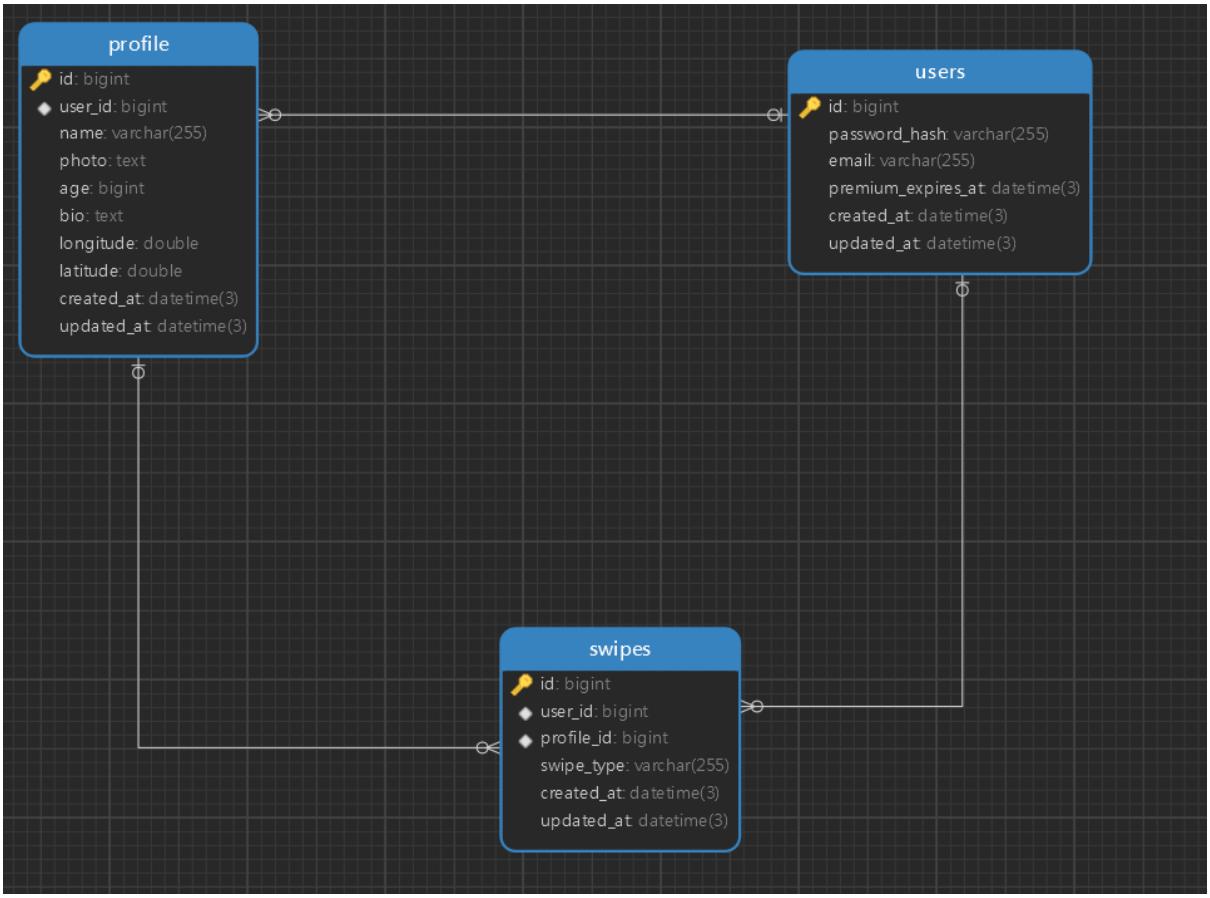
    d. Go Standard Project Layout : The Go Standard Project Layout is chosen to ensure a consistent and well-organized structure for the codebase. This layout follows widely accepted conventions in the Go community, making it easier for developers to navigate and understand the project. By adhering to this standard, the project benefits from improved maintainability, scalability, and collaboration, as developers can quickly familiarize themselves with the project's structure. This layout also facilitates the application of best practices and simplifies the integration of tools and libraries commonly used in the Go ecosystem.

    e. Database MySQL : MySQL is chosen for its reliability, stability, and robustness as a widely-used relational database. It excels at handling structured data and complex queries. MySQL's support for ACID (Atomicity, Consistency, Isolation, Durability) properties ensures data integrity, which is crucial for transactional applications like a dating app. Additionally, MySQL's ability to scale both vertically and horizontally makes it a suitable choice for applications that need to handle increasing amounts of data and users.

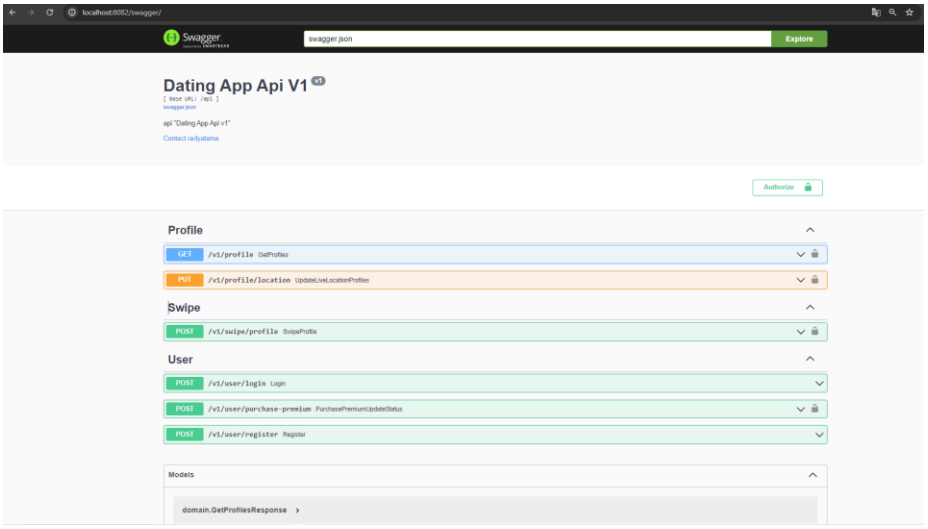
    f. Redis Cache : Redis is selected for its exceptional performance as an in-memory data structure store, offering fast read and write operations. This makes it ideal for

caching frequently accessed data, reducing the load on the database and improving response times. Redis's support for complex data types such as strings, hashes, lists, sets, and sorted sets enables the implementation of various caching strategies. Additionally, Redis is well-suited for session management, thanks to its fast access speeds and ability to handle large volumes of data with low latency. Its Pub/Sub feature further enhances its utility, making it useful for implementing real-time features like notifications and messaging within the app.

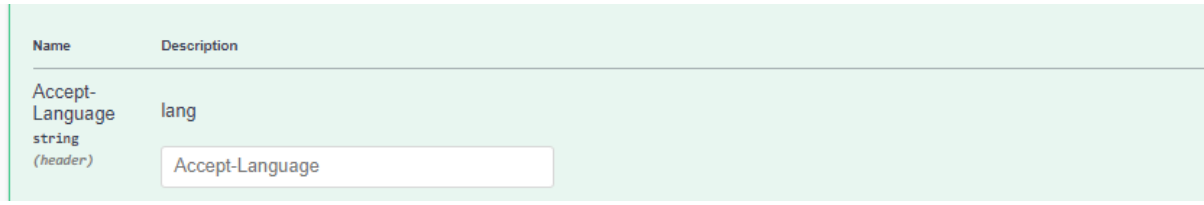
4. ERD Diagram :



5. Test:
    a. Open the app swagger http://localhost:8082/swagger/



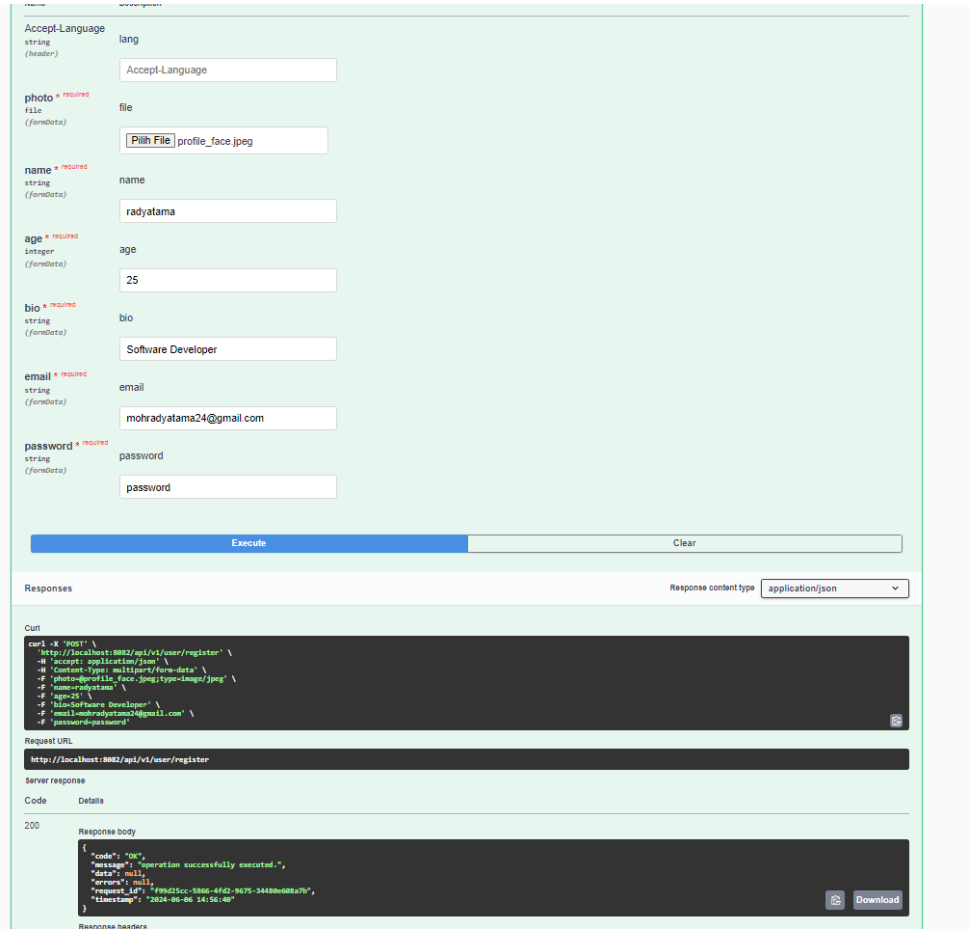
The Accept-Language to manage reponse language, by default is `english`, for available is `en/id` , en = English, id = Bahasa.



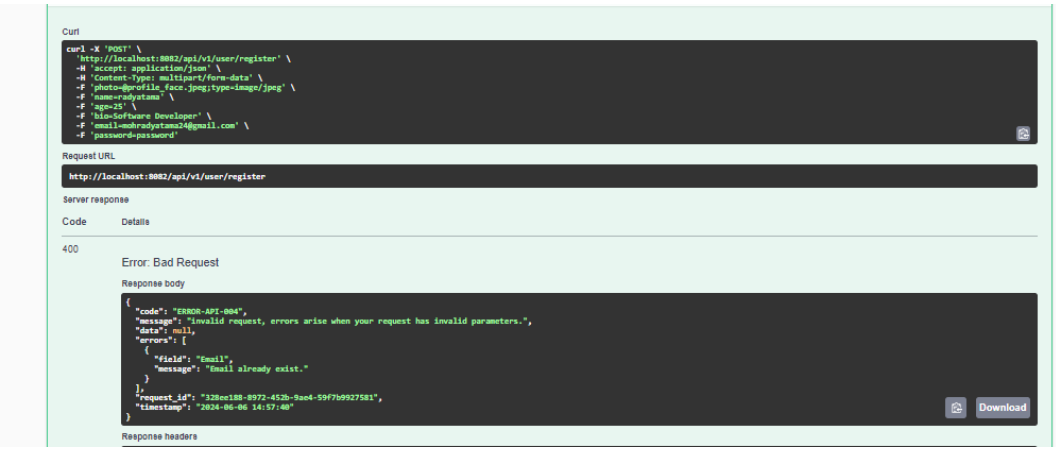
Example for lang id

b. Register your account

- success



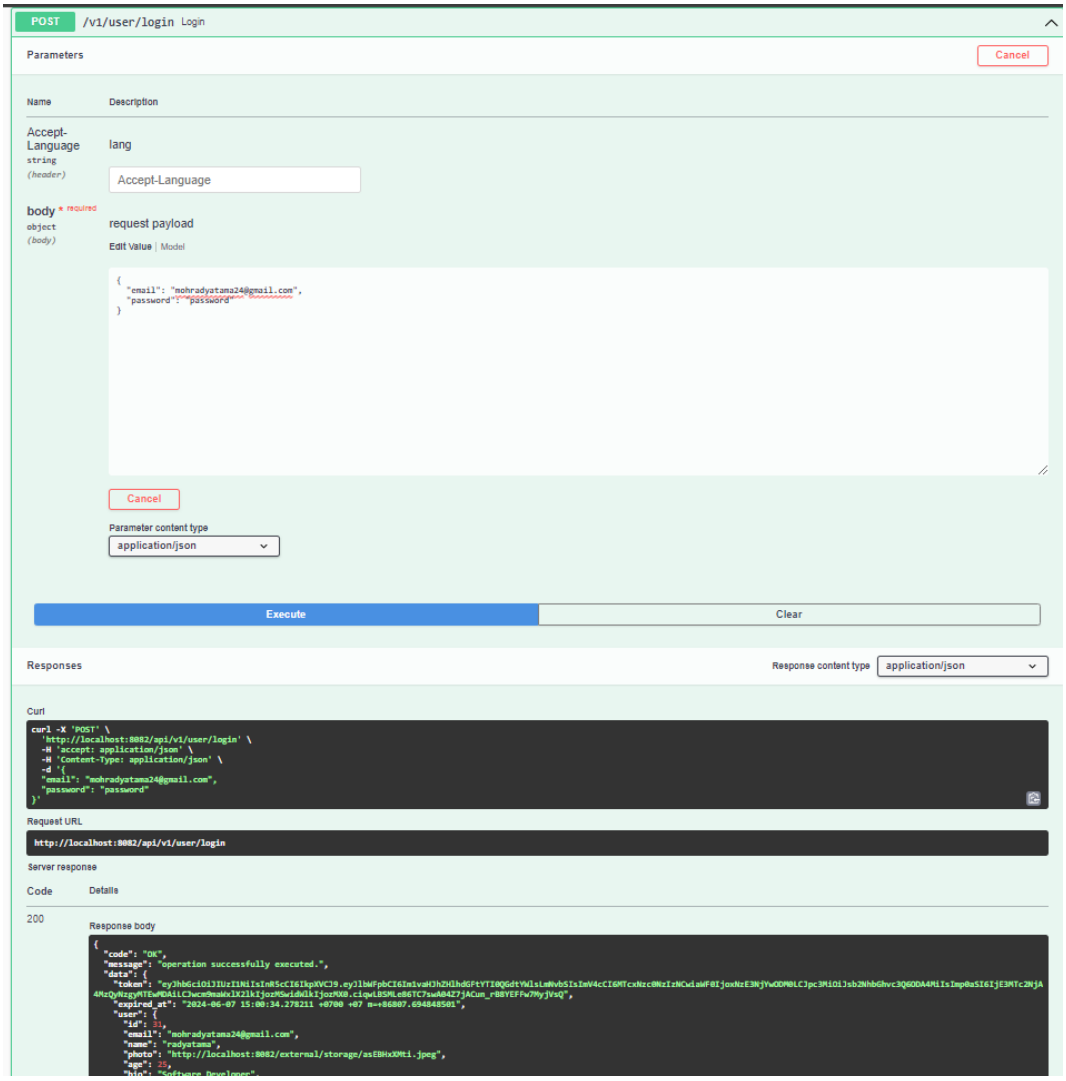
- error user already exist



- error validation

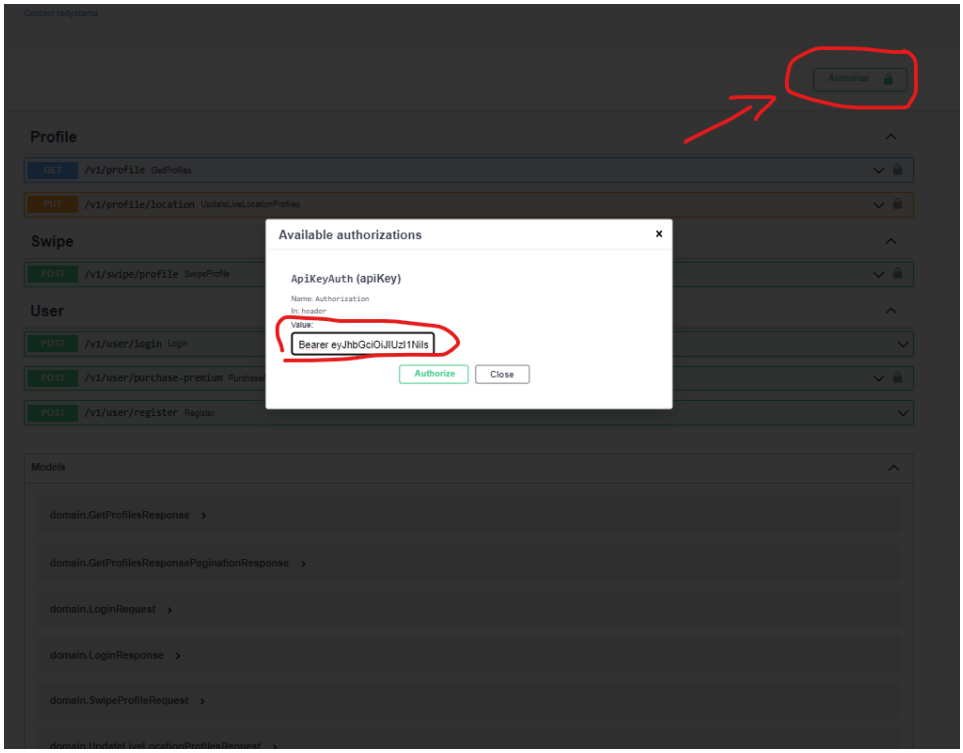
c. Login User :

- success



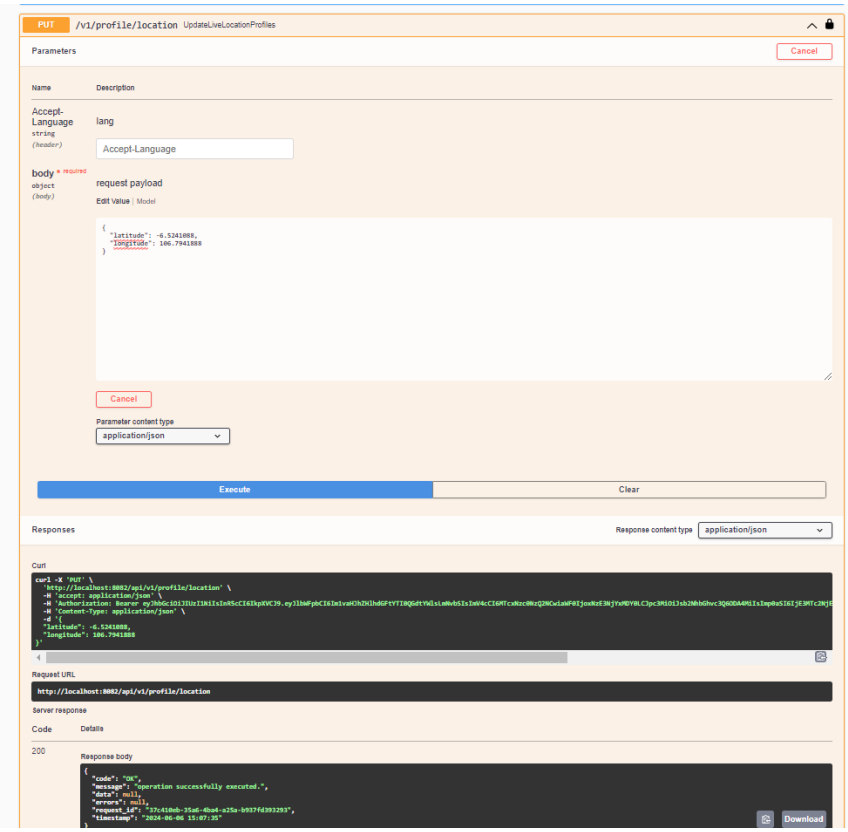
- error invalid email password

d. Put your token with Bearer <Token> in the Authorize swagger then click Authorize then close



e. Update current your location

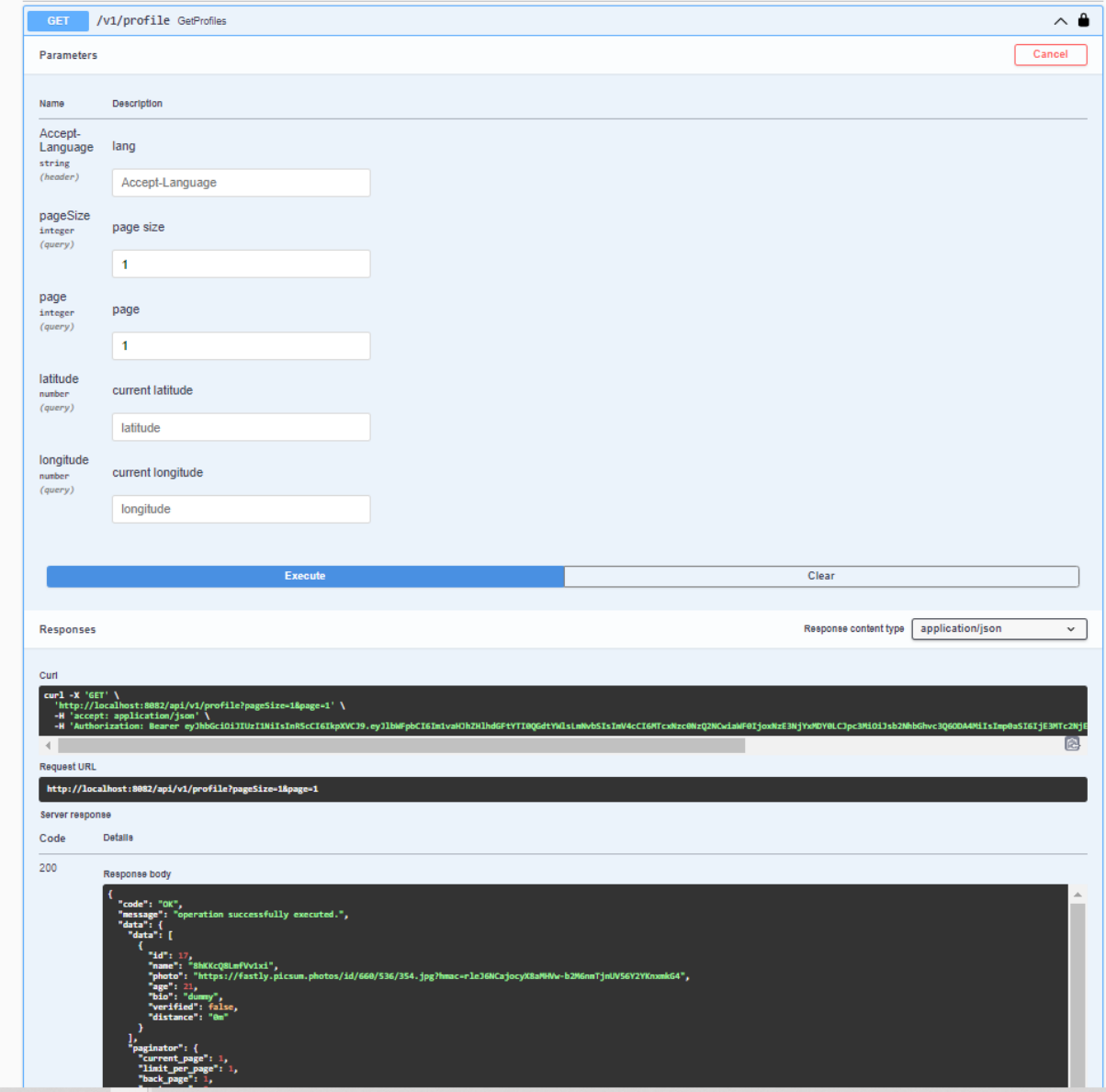
- success



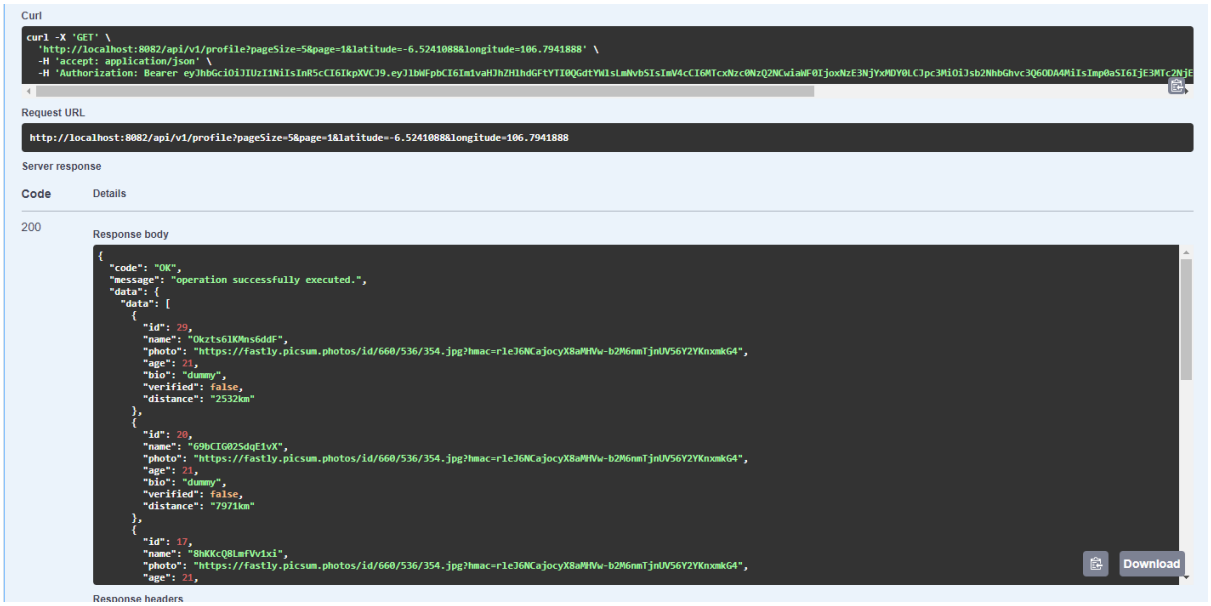
- error token

f. Get profiles list

- With pagination

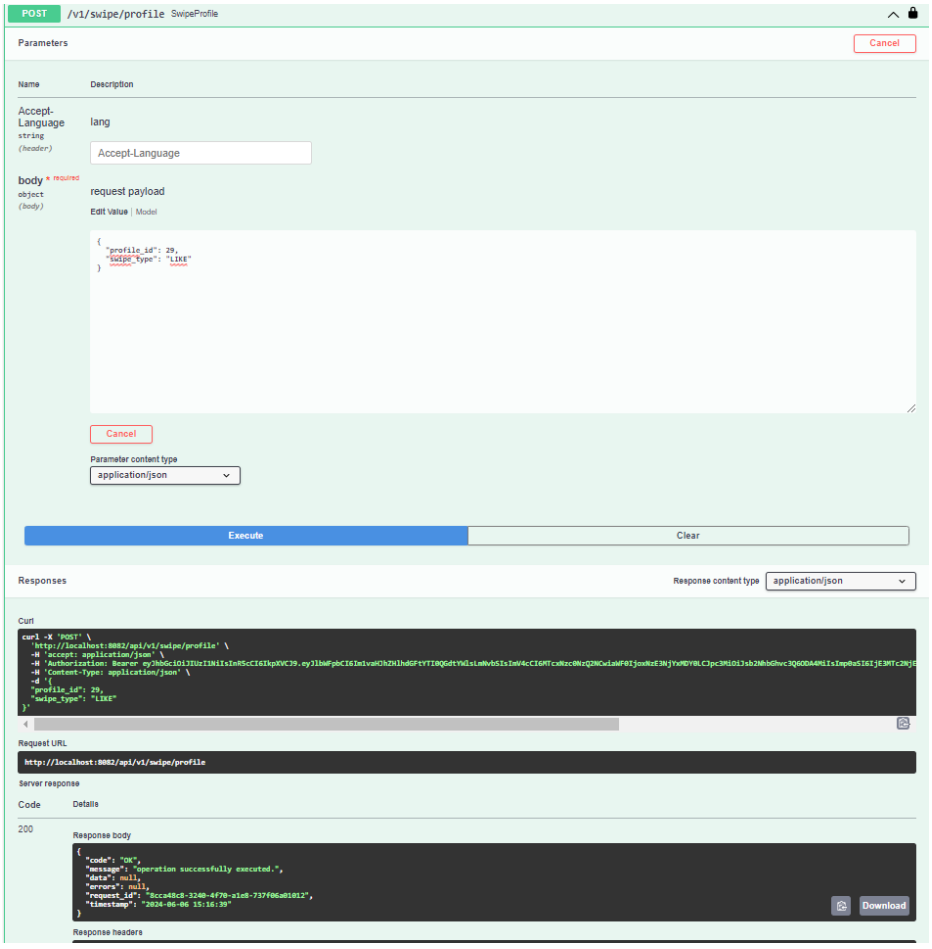


- Pagination with long lat request nearby

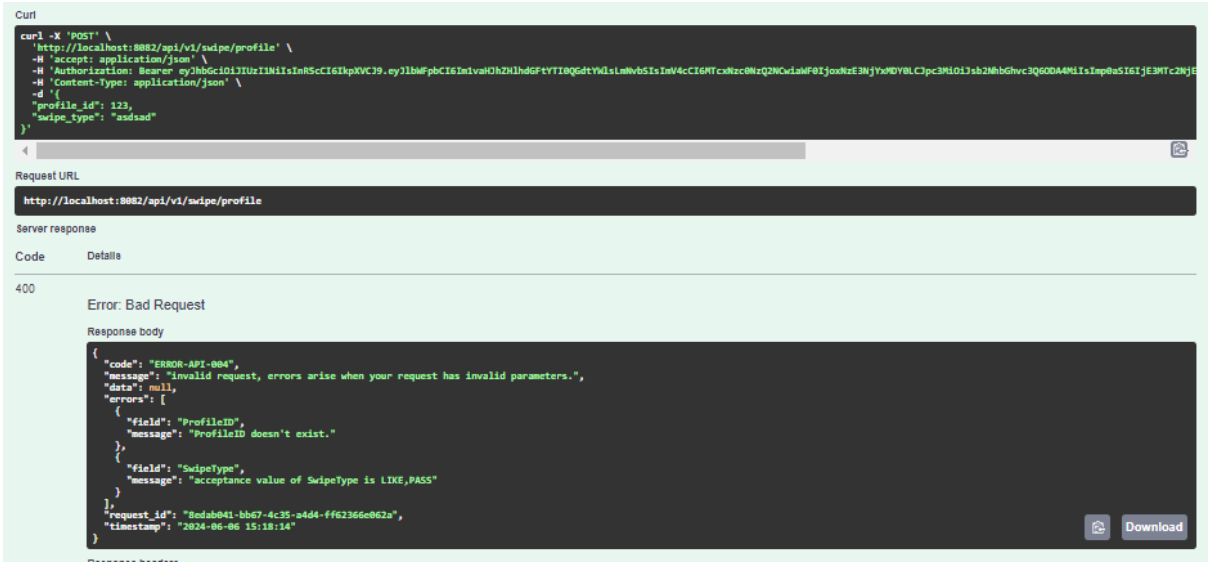


g. Swipe profile , `profile_id` is from `id` get profile list, swipe_type can fill (LIKE/PASS)

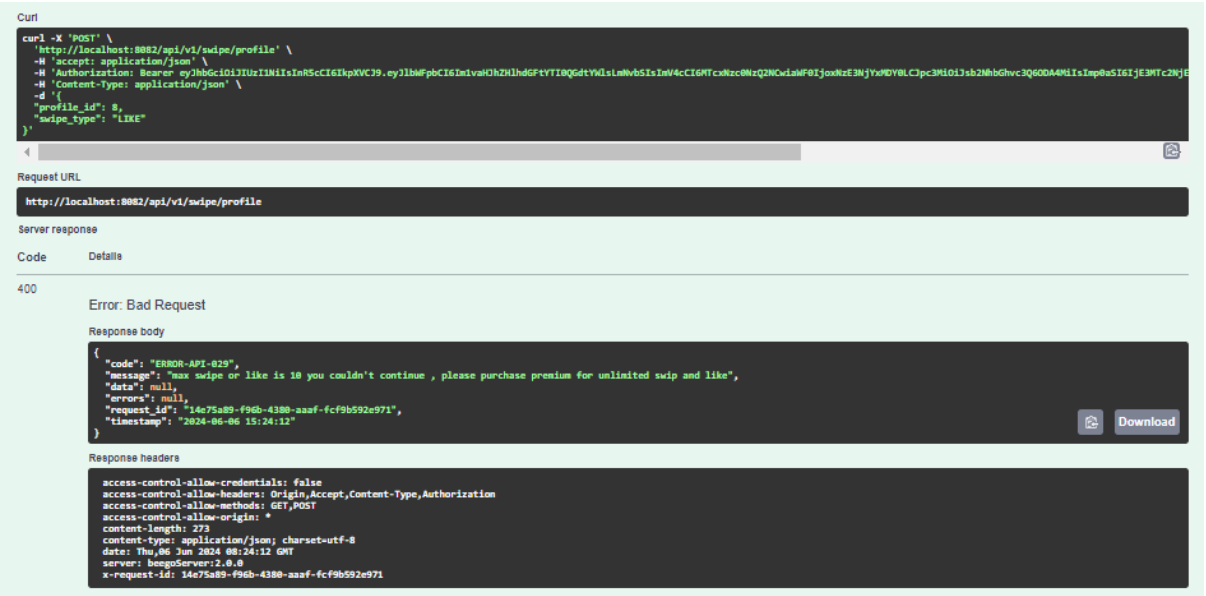
- Success



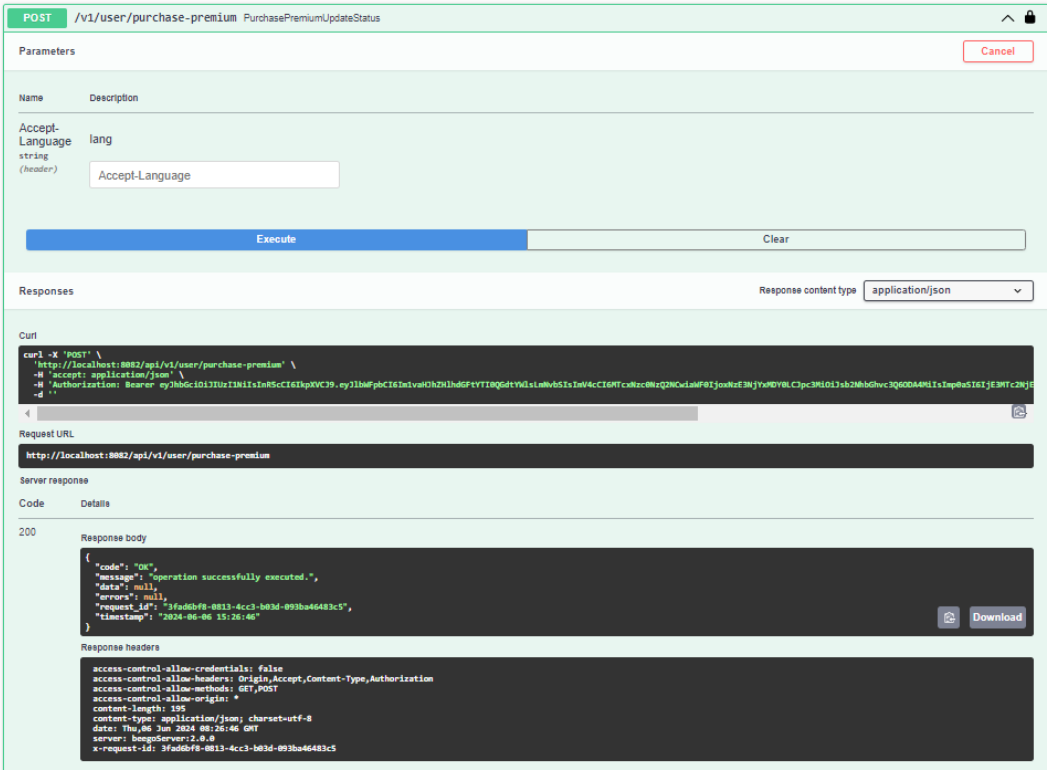
- Error validation



- Error swipe pass/like quota has run out max 10 profile (like + pass) per day



h. Purchase premium , this is a callback api for the payment gateway when it is successful it will hit this api. The premium will be active as monthly



Then the account has been verified