Basic Genuin Platform API

Description :

We are building a platform that allows users to interact with brands, communities, and videos in a variety of categories. Your task is to design the backend for the application using Node.js, Express, and a relational database of your choice (MySQL, PostgreSQL, or any other relational DB). Additionally, caching mechanisms such as Redis or Memcached should be implemented for performance optimization.

Time Frame

- Total Duration: 2 days
- Recommended breakdown:
    - Backend Development: 1 day
    - Testing & Documentation: 1 day

Technical Stack Requirements :

- **Backend (Choose one):**
    - Node.js with Express
    - Golang with Gin/Echo

- **Database:**
    - Mysql or Postgres
    - Redis or Memcache for caching

Core Features to Implement

Major Tables Required Includes:

Users: which has phone number or email address. Must: Username(Unique)
Optional age, gender, DP, country,etc.
Brand: has brand name, website url, primary & secondary color, thumbnail url, Username, etc.
Community: Belongs to Brand. Has name, thumbnail, DP, Members (which are users)
Groups: Belongs to Communities. Has name, Members, Video counts,
Videos: Belongs to Groups. Created by users.

Data Population:

1. Populate at least 200 user records.
2. Populate at least 20 brand records.
3. Populate at least 50 community records.
4. Populate at least 100 group records.
5. Populate at least 400 group members.
6. Populate at least 100 video records.

Backend Requirements

1. Authentication System
    a. JWT-based authentication
    b. Role-based access control
    c. Session management
2. Genuin Basic Core
    a. CRUD operations for brands/communities/groups/videos
    b. Submission validation
3. User System
    a. Signup/Login
    b. Join community/Join Group
    c. Create Video
    d. Add members to group/community via username(unique)

Caching:

Use a caching system like **Redis** or **Memcached** to improve performance for frequently accessed data. The following data should be cached:

- Recent videos
- Recent groups
- Top videos by views
- Top brands with the most video views

Other API requirements

/brands/videos/top : Get Retrieve a list of brands sorted by the highest video views, in descending order
/brands/videos/top?brand_id={brand_id}: Retrieve the videos with the most views for a specific brand

/videos/recent?brand_id={brand_id}: Retrieve the most recent videos for a specific brand (brand_id is optional)
/videos/highlights?brand_id={brand_id}: Retrieve the most discussed (commented) video for a given brand

Technical Expectations

Code Quality

- Clean code principles
- SOLID principles
- Design patterns usage
- Proper error handling
- Input validation
- Security best practices

Testing

- Minimum 80% test coverage
- Unit tests
- Integration tests
- API tests
- Frontend component tests

Documentation

- API documentation (Swagger/OpenAPI)
- Setup instructions
- Database schema
- Architecture diagram
- README file

Submission Requirements

Deliverables

1. GitHub repository link
2. Setup documentation
3. API Documentation
4. Test coverage report
5. 5-minute demo video
6. Database schema diagram

Evaluation Criteria

- Code Quality: 30%
- Feature Implementation: 25%
- Testing: 20%
- Documentation: 15%
- Security: 10%

Bonus Points:

- Implement rate limiting for APIs to prevent abuse.
- Implement pagination where necessary (e.g., for large lists of brands, videos, etc.).
- Write tests for your API endpoints using tools like Mocha, Chai, or Jest.

Submission Process

1. Create a private GitHub repository
2. Implement the solution
3. Record a demo video
4. Share repository access with submissions@begenuin.com
5. Send completion email to submissions@begenuin.com
6. The solution must be zipped and shared via email to submissions@begenuin.com and should not be shared through any ***public cloud storage***.

Good luck! We're excited to see your solution.

BeGenuin Technical Team