API Integration & User Management Assignment

Objective

Your task is to build a Node.js API that integrates with external APIs, manages users, and implements authentication & authorization. The assignment evaluates your ability to consume external APIs, handle authentication, process data, and structure a maintainable Node.js service.

Requirements

1. User Authentication & Management

Implement a user management system with JWT-based authentication.

User Model:

- id (UUID)
- name (string)
- email (unique, string)
- password (hashed, string)
- role (enum: admin, user)
- createdAt, updatedAt (timestamps)

Authentication API (JWT)

- POST /auth/register -> Register a new user
- POST /auth/login -> Log in and receive a JWT token
- GET /auth/profile -> Get the authenticated user's profile (Requires Auth)

User Management API (Admin Only)

- GET /users -> List all users
- GET /users/:id -> Get user by ID
- PATCH /users/:id -> Update a user
- DELETE /users/:id -> Delete a user

2. API Integration Task

Fetch real-time weather data and cryptocurrency prices, process them, and expose an API endpoint.

Implementation

- Use OpenWeatherMap API to fetch weather data.
- Use CoinGecko API to fetch cryptocurrency prices.
- Combine data from both sources into a unified response.

Endpoints

- GET /data?city=London&currency=bitcoin

  Example Response:

```
{
  "city": "London",
  "temperature": "15°C",
  "weather": "Cloudy",
  "crypto": { "name": "Bitcoin", "price_usd": 42000 }
}
```

- POST /data -> Accepts parameters to modify how data is processed.

3. Error Handling & Caching

- Handle errors: Invalid API responses, rate limits, network failures.
- Implement retries: Retry failed API calls up to 3 times.
- Cache responses: Store API responses in memory for 5 minutes.
- Force refresh: Support ?refresh=true to bypass cache.

4. Best Practices

- Use environment variables (.env) for API keys & configurations.
- Use Express or Fastify.
- Use async/await and modern ES6+ syntax.
- Follow a structured folder layout.
- Implement unit tests (Jest or another framework).
- Document API endpoints in Swagger or README.

Evaluation Criteria

- Code Quality -> Readability, maintainability, modularity.
- API Integration -> Handling authentication, rate limits, error handling.
- Security -> Proper JWT authentication, password hashing, role-based access.
- Performance -> Caching, retries, structured logging.

- Testing -> Unit tests covering key functionalities.

Deliverables

- GitHub repository with the complete code.
- A README.md file explaining:
  - How to install & run the project.
  - Required API keys & where to obtain them.
  - Example API requests & responses.

Submission Deadline

You have **7 days** to complete this assignment.

Good luck!