Objective:

Build a backend system for a simplified **Smart Device Management Platform** where users can register, authenticate, and manage their devices.

Assignment Brief

You are required to **design and implement a backend service** that provides APIs for the following:

1. User Management

Endpoints

- **POST /auth/signup** → Create account.
- **POST /auth/login** → Login with email & password, return JWT.

Sample Payload (Signup)

```json
{
  "name": "John Doe",
  "email": "john@example.com",
  "password": "SecurePass123",
  "role": "user"
}
```

Sample Response (Signup)

```json
{
  "success": true,
  "message": "User registered successfully"
}
```

Sample Payload (Login)

```json
{
  "email": "john@example.com",
  "password": "SecurePass123"
}
```

Sample Response (Login)

```json
{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "u1",
    "name": "John Doe",
    "email": "john@example.com",
    "role": "user"
  }
}
```

---

2. Device Management

Endpoints

- **POST /devices** → Register new device.
- **GET /devices** → List devices (filter by `type`, `status`).
- **PATCH /devices/:id** → Update device details.
- **DELETE /devices/:id** → Remove device.
- **POST /devices/:id/heartbeat** → Update `last_active_at`.

Sample Payload (Register Device)

```json
{
  "name": "Living Room Light",
  "type": "light",
  "status": "active"
}
```

Sample Response (Register Device)

```json
{
  "success": true,
  "device": {
    "id": "d1",
    "name": "Living Room Light",
    "type": "light",
    "status": "active",
    "last_active_at": null,
    "owner_id": "u1"
  }
}
```

Sample Payload (Heartbeat)

```json
{
  "status": "active"
}
```

Sample Response (Heartbeat)

```json
{
  "success": true,
  "message": "Device heartbeat recorded",
  "last_active_at": "2025-08-17T10:15:30Z"
}
```

3. Data & Analytics

Endpoints

- **POST /devices/:id/logs** → Create log entry.
- **GET /devices/:id/logs?limit=10** → Fetch last 10 logs.
- **GET /devices/:id/usage?range=24h** → Aggregated usage.

Sample Payload (Log Entry for Smart Meter)

```json
{
  "event": "units_consumed",
  "value": 2.5
}
```

Sample Response (Last 10 Logs)

```json
{
  "success": true,
  "logs": [
    {
      "id": "l1",
      "event": "units_consumed",
      "value": 2.5,
      "timestamp": "2025-08-17T08:00:00Z"
    },
    {
      "id": "l2",
      "event": "units_consumed",
      "value": 1.2,
      "timestamp": "2025-08-17T09:00:00Z"
    }
  ]
}
```

Sample Response (Aggregated Usage)

```json
{
  "success": true,
  "device_id": "d2",
  "total_units_last_24h": 15.7
}
```

4. Advanced Requirements (Bonus)

- **Rate limiting** (100 requests/min per user).
- **Background job**: auto-deactivate device if `last_active_at` > 24h.
- **Unit tests** (Jest/Mocha).
- **Dockerized setup**.

Technical Requirements

- **Node.js (Express/NestJS/Fastify)**.
- **PostgreSQL or MongoDB**.
- **JWT authentication**.
- **Clean architecture** (controllers, services, models).
- **Validation** (e.g., Joi, Zod).
- Provide a **Postman collection**.

Submission Guidelines

- Push code to **GitHub repo** with proper commits.
- Include `README.md` with:
  - Setup instructions.
  - API documentation.
  - Any assumptions made.
- Deadline: **72 hours**.

Evaluation Criteria

- Functional correctness.
- Code structure & maintainability.
- Security & validation.
- Efficient DB schema design.
- Handling of edge cases.
- Bonus points for scalability, tests, Docker, CI/CD.