Here's a clear and simplified summary of what you need to do for the Distributed Energy System Project — structured for easier understanding and implementation.

🔧 Project Goal

Simulate a distributed energy community using 6 independent microservices, exchanging data via RabbitMQ, and interacting via a REST API + JavaFX GUI.

🧱 Core Components (Services You Must Build)

1. Community Energy Producer

- Sends "PRODUCER" messages every few seconds.
- Includes:
  - type: PRODUCER
  - association: COMMUNITY
  - kwh: random value (e.g. 0.003)
  - datetime: current timestamp
- Should simulate solar/weather data for realistic values.

2. Community Energy User

- Sends "USER" messages every few seconds.
- Includes:
  - type: USER
  - association: COMMUNITY
  - kwh: random value (e.g. 0.001)
  - datetime: current timestamp
- Simulate more energy use in peak hours (morning/evening).

3. Usage Service

- Listens to producer/user messages.
- Updates the hourly aggregated database table:

```sql
hour | community_produced | community_used | grid_used
```

- Calculates grid usage if community energy is not enough.
- Sends an **update message** when new data is saved.

4. Current Percentage Service

- Listens to **update messages**.
- Calculates and updates:

```sql
hour | community_depleted | grid_portion
```

- E.g., if community is depleted → 100%, grid used → %.

5. Spring Boot REST API

- Provides two endpoints:
  - GET /energy/current → returns current percentage data.
  - GET /energy/historical?start=...&end=... → returns usage data in a given range.
- **Reads from DB only** (doesn't modify it).
- Built using **Spring Boot**.

6. JavaFX GUI

- Sends HTTP requests to the REST API.
- Displays:
  - **Current status** (percentage, grid use).
  - **Historical usage** (filtered by time).
- GUI is **not directly connected** to the database.

✅ Final Submission Requirements

To get full marks, your project must:

- Be **runnable independently** for each service.
- Use:
  - **Spring Boot** for REST API
  - **JavaFX** for GUI
  - **RabbitMQ** for service communication
- Contain a working **GitHub repo link**.
- Have **clean code**, a good presentation, and you must be able to **explain it**.

📈 Grading Breakdown

Mid-Project Milestone (Class 8 – 50% total)

- REST API: Returns mock data.
- GUI: Fetches & displays mock data from REST API.

Final Project (50% total)

- ✅ REST API (real DB reads): 10%
- ✅ Energy Producer (sends messages): 10%
- ✅ Energy User (sends messages): 10%
- ✅ Usage Service (process + update DB): 40%
- ✅ Current Percentage Service: 30%

🕒 Example Data Flow (Important!)

1. **Producer/User** sends message →
2. **Usage Service** updates DB →
3. **Sends update message** →
4. **Current % Service** updates % table →
5. **REST API** returns data →
6. **GUI** displays it to user.
