Please read through everything and follow the steps. We cannot help you technically as it's required to solely finish the task without external help to make it equally fair for every participant.

BACKEND ENGINEERING TASK
ESTIMATED TIME: 480 MINUTES (you are allowed to take more time, but please plan a full working day)

Your task is to generate an invoice and daily sales Report System

Required Skills: NestJS, MongoDB, RabbitMQ, Cron Jobs, RESTful APIs, Email Sending, Testing (Unit and Integration)

Scenario

Your task is to build a system where users can create invoices for sales transactions. The system will automatically generate a daily sales summary report and send it via email at 12:00 PM each day. The report will be sent through a RabbitMQ queue, where a separate consumer service will process the email sending.

Task Breakdown

1. Invoice Creation Service:

- **Project Setup:**
  - Initialize a NestJS project.
  - Integrate MongoDB for data persistence.

- **MongoDB Schema Design:**
  - Design the Invoice model based on the fields provided:
    - **customer** (string): The name or identifier of the customer.
    - **amount** (number): The total amount of the invoice.
    - **reference** (string): A reference code for the invoice.
    - **date** (date): The date the invoice was created.
    - **items** (array of objects): Each item includes:
      - sku (string): Stock Keeping Unit, unique identifier for the item.
      - qt (number): Quantity of the item.

- **REST API Development:**
  - Develop the following REST endpoints:
  1. POST /invoices - Create a new invoice.

2. GET /invoices/:id - Retrieve a specific invoice by ID.
3. GET /invoices - Retrieve a list of all invoices (with optional filters like date range).

- **Testing:**
    - Write unit tests for the invoice creation and retrieval logic.
    - Write integration tests for the REST API endpoints using supertest.

2. Daily Sales Summary Report:

- **Cron Job Setup:**
    - Implement a cron job in the Invoice Service to run daily at 12:00 PM.
    - The cron job should:

1. Calculate the total sales for the day.
2. Calculate the total quantity sold per item (grouped by SKU).
3. Prepare a summary report.

- **RabbitMQ Integration:**
    - Publish the daily sales summary report to a RabbitMQ queue named daily_sales_report.
    - The message should include:

1. Total sales amount.
2. Per item sales summary (SKU, total quantity sold).

- **Consumer Service for Email Sending:**
    - Create a separate NestJS service (email-sender) to consume messages from the daily_sales_report queue.
    - Implement the consumer to:

1. Receive the sales summary report.
2. Send an email containing the report (you can mock the email sending or use a service like SendGrid).
3. Write unit tests for the message processing and email sending logic.

Technical Requirements

Invoice Creation Service:

- Proper MongoDB schema and REST API for invoice management.
- Cron job for daily sales report generation.
- RabbitMQ producer to publish sales summary reports.

Consumer Service (Email Sending):

- RabbitMQ consumer to process sales summary reports.
- Mock or real implementation for sending emails with the sales report.

General:

- Use Docker or Docker Compose for easy setup and running both services together.
- Ensure proper error handling and logging in both services.
- Testing should cover critical functions and edge cases.

Deliverables

- **ZIP File Submission:**
    - Compress the project files into a single .zip file.
    - The ZIP file **must not** include the following directories:
        - .git (Git version control folder)
        - node_modules (Dependencies)
        - dist (Compiled output)
    - The structure inside the ZIP file should be clean and ready to run after extraction.

- **README File:**
    - Provide a README.md file within the ZIP that includes:
        - Setup instructions.
        - How to run the services.
        - Any additional notes on the implementation.

Evaluation Criteria

- **Correctness:** The solution should correctly create invoices, generate daily reports, and send emails.

- **Microservice Communication:** Proper use of RabbitMQ for passing reports between services.
- **Testing:** Comprehensive test coverage for both services.
- **Cron and Email Integration:** Effective use of cron jobs and handling of email sending.
- **Submission Quality:** The ZIP file should be correctly structured and contain all necessary files as per the instructions.

The main goal of the task is to show your skills in the best way possible.