AdventureWorks Product Review App

Description: Build an online product review feature. The feature will accept product reviews, scan the review for any inappropriate language, post the review live once approved or archive if not approved and notify the reviewer via email (simulated) once the review status has been finalized.

Required components:

- API Layer (HTTP)
- Database (Postgres or MySQL) using the AdventureWorks for Postgres or MySQL
- Worker Layer #1 - Review processor
- Queue, Messaging Bus
- Worker Layer #2 - Notifications
- Reports/Queries

Requirements, Components and Acceptance criteria:

- API Layer (HTTP):

  - Client submits a product review via HTTP:

    ```
    curl -X POST http://0.0.0.0:8888/api/reviews \
        -H 'Content-Type: application/json' \
             -d '{
                "name": "Elvis Presley",
                "email": "theking@elvismansion.com",
                "productid": "8",
                      "review": "I really love the product and will
                recommend!"
            }'
    ```

  - API HTTP response:

    ```
    {
        "success": true,
        "reviewID": [id integer]
    }
    ```

    The API puts the product review into the database and onto a queue for processing.

- AdventureWorks Database – MySQL or Postgres:
    - https://github.com/lorint/AdventureWorks-for-Postgres
    - https://sourceforge.net/projects/awmysql/files/latest/download
    - Create necessary database changes to support this workflow

- Worker #1 – Review Processor
    - Worker consumes a review from the queue and scans for inappropriate language
    - Bad words that should be filtered are: **BAD_WORDS = ["fee", "nee", "cruul", "leent"]**. If a message contains a bad word, the message should be marked as inappropriate.

- Queue, Message Bus
    - Queuing and messaging mechanism for review processor (as needed) (Redis, RabbitMQ, etc.)
    - Messages should be queued from the API to Worker #1 and Worker #2

- Worker #2 - Notification:
    - Once the review is approved and published, the reviewer is notified via email.
    - The worker should fetch the relevant data to process the notification but the notification is purely simulated (no email is actually sent).

Configuration notes:

Please create a README that includes app setup instructions for all components. This could include:

- Bash or other shell scripts to build/run
- sql files
- Docker files (if used)
- Web page(s) if available
- Path to logs

- Pre-requisites (runtimes, libs, etc.)

Note, this does not have to be a polished README but should get a savvy reader running the app.

Publishing:

Please provide link to Git repository. If the API is deployed online, please provide cURL script or command to test.

Tips

1. Be aware that you may encounter some "issues" with the data in the AdventureWorks database that you may have to fix. This is expected. Please make note of these issues.
2. For queuing, please we favor Redis but please use whatever queue framework you are familiar.
3. Ideally, we should be able to take the completed project and run it locally via scripts. Our systems are Ubuntu Linux.
4. Please create a README that gives me some instructions on how to build and run the API and related components. The README does not have to be fancy. A simple check list will be fine. You can safely assume that those who review the README and attempt to run your app are developers.
5. If you don't finish all the work, this does not mean you are disqualified. We will review how much was completed and the quality of the work.
6. Prioritize a complete and working project over all other considerations
7. Please make note of any questions or issues you encounter. We will certainly take these into consideration.

Bonus Points!

1. All component services (API, database, queues) running in Docker
2. Database migrations as needed
3. Unit tests as needed
4. Deployment to AWS or any public cloud
5. API security