Gardrops Image Upload Session Micro-services (Spring Boot) Assignment

Objective

Create an image upload pipeline composed of two small Spring Boot microservices: a public API for session management and an internal service for image transformation and storage.

Scenario

A client must let users upload up to **10 images** within a short-lived session, where each image is resized/compressed before being stored.

Service	Purpose	Port
ImageUploadApi	Public-facing REST API that creates/controls sessions and hands raw images off for processing.	**8080**
ImageProcessingApi	Internal REST API that performs image transformations and writes the resulting JPEGs to disk.	**8081**

Functional Requirements

1 — ImageUploadApi

Endpoint	Requirements	Returns
POST /sessions	Create a new upload session (1-hour TTL).	created `sessionId`
POST /sessions/{sessionId}/images	Validate session exists & not expired. Validate max image count. Forward to processing service.	created `imageId`
DELETE /sessions/{sessionId}/images/{imageId}	Validate session exists & not expired. Validate image is present in the session. Delete file from storage and session.	-
GET /sessions/{sessionId}/images	List all imageIds currently linked to the session.	`imageIds` in the sesssion

Additional rules

- Each session has **1-hour lifetime** (from the moment session is created)
- Each session may hold **max 10 images**.
- Session & image IDs must be **UUIDs**.
- Each endpoint are throttled to **10 requests/minute per IP**.
- Session data and rate-limit counters are stored in **Redis**.

2 — ImageProcessingApi

Endpoint	Requirements	Returns
POST /images/process	Resize multipart image to ≤ 720x1280px Compress to 90% JPEG Save to destination path	-

- Accessible **only from localhost (127.0.0.1)**; no rate limiting.
- May use any Java image library (`javax.imageio` recommended).

cURL list

```
curl -X POST localhost:8080/sessions
```

```
curl -X POST localhost:8080/sessions/{sessionId}/images -F image=@image.jpg
```

```
curl -X DELETE localhost:8080/sessions/{sessionId}/images/{imageId}
```

```
curl -X GET localhost:8080/sessions/{sessionId}/images
```

```
curl -X POST localhost:8081/images/process -F image=@"orig.jpg" -F destinationFilePath="{imageId}.jpg"
```

Technical & Architectural Expectations

- **Language:** Kotlin preferred, Java accepted.
- **Framework:** Spring Boot, Spring Data Redis (or Lettuce), Spring Validation.
- **Authorization**: No need
- **Persistence:** Temp file storage may be the local filesystem; structure it under `/tmp/uploads/{sessionId}/`.
- **Rate limiting:** Implement with Redis
- **Session expiry:** Rely on Redis TTL
- **Request/response:** Accept proper inputs in the request, return proper outputs in the response.
- **Project Structure:** Follow clean, layered packaging (controller → service → repository).

Deliverables

1. **Source code repository (Git)** containing both Spring Boot projects
2. (Optional) **Postman collection**

Evaluation Criterias

Area	What we look for
Correctness	All requirements met & edge cases behave as specified.
Code Quality	Readability, folder structure, meaningful naming
API Design	RESTful principles, clear contracts, proper status codes & validation.
Architecture	Clean layering, Redis usage, configuration management

---

Submission Instructions

- **Deadline:** 36 hours after the assignment is sent.