Assignment 2

Detailed Requirements Specification, Client-Server Architecture & High-Level Design

Image Editor (Cloud-Ready)

Student One (Ashoka ID: 12345)
Student Two (Ashoka ID: 67890)

Department of Computer Science, Ashoka University

October 27, 2025

Contents

1 Introduction **2**

2 Detailed Requirements Specification **2**
2.1 Functional Requirements . . . 2
2.2 Non-Functional Requirements . . . 2
2.3 Error Handling Matrix . . . 2

3 Client-Server Architecture (Cloud-Native) **3**
3.1 Topology Diagram . . . 3
3.2 Component Responsibilities . . . 3
3.3 Security & Compliance . . . 3
3.4 DevOps Observability . . . 4

4 High-Level Design **4**
4.1 Core Abstractions . . . 4
4.2 Synchronous Flow Diagram . . . 5
4.3 Async Batch Flow . . . 5
4.4 Scalability Notes . . . 6
4.5 Extensibility Hooks . . . 6

5 Conclusion **6**

1 Introduction

This document fulfils Assignment 2 for CS-3810. We specify a cloud-deployable Image Editor that allows users to upload, transform, and download images through a serverless architecture. All design is conveyed textually and via inline TikZ diagrams.

2 Detailed Requirements Specification

2.1 Functional Requirements

ID	Requirement Statement	Type	Pri	Acceptance Criteria
F-01	Upload JPEG/PNG up to 10 MB	Func	Must	HTTP 201, persisted, URL returned
F-02	Rotate 90° CW/CCW	Func	Must	Pixel-perfect (SSIM ≥ 0.99)
F-03	Flip Horizontal / Vertical	Func	Must	Same quality gate as F-02
F-04	Resize with locked aspect (200–4000 px)	Func	Must	Lanczos-3, $\leq$ 500 ms p95
F-05	Compress quality 10–100 %	Func	Must	File size $\leq$ original
F-06	Generate 400 px WebP thumbnail	Func	Should	$\leq$ 300 ms, CDN cached
F-07	Undo last operation instantly	Func	Should	State reverts, no extra cost
F-08	Batch apply op to ≥ 5 images	Func	Could	Async job + e-mail link
F-09	Download result PNG/JPEG/WebP	Func	Must	Correct MIME, 200 OK
F-10	Auto-delete after 24 h	Func	Must	S3 lifecycle or cron + TTL

2.2 Non-Functional Requirements

NF-01 99.9 % monthly uptime (GCP Monitoring).

NF-02 p95 transform latency $\leq$ 500 ms under 5 concurrent users/region.

NF-03 GDPR compliant – AES-256 at rest, TLS 1.3 in transit.

NF-04 Auto-scale 0 $\rightarrow$ 20 container instances (CPU 60 %).

NF-05 Support 1 000 concurrent users with CDN edge caching.

2.3 Error Handling Matrix

Error Scenario	HTTP	End-User Message	Server Action
File size $>$ 10 MB	413	“Image too large (max 10 MB)”	Reject, log metric

Unsupported MIME	415	"Only JPG and PNG allowed"	Same
Corrupt image	422	"Cannot decode image"	Increment bad_image_total
Transform timeout (> 10 s)	504	"Try again later"	Dead-letter queue
Storage quota exceeded	507	"Temporarily full"	Trigger early cleanup

3 Client-Server Architecture (Cloud-Native)

3.1 Topology Diagram

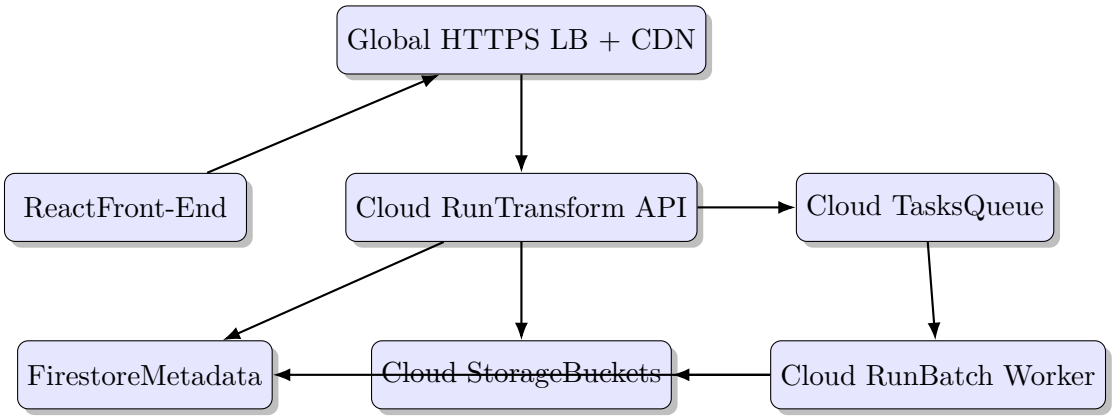

Figure 1: High-level cloud topology (all serverless, auto-scaling)

3.2 Component Responsibilities

Front-End
React + TypeScript, Vite-bundled, hosted on Firebase Hosting. Drag--drop UI, canvas preview, axios client.

API Gateway
Google Cloud Endpoints; 20 req/s per-IP rate limit.

Transform Service
Node.js 20 container with Sharp (libvips); CPU-bound ops in memory.

Object Storage
Two buckets: `uploads-raw` and `results`. Signed URLs (1 h TTL).

Metadata Store
Firestore doc: `{imageId, ops[], ttl}`. TTL triggers cleanup.

Batch Worker
Separate Cloud Run service polled by Cloud Tasks; compresses/rotates in parallel, creates zip, mails link.

3.3 Security & Compliance

- VPC Service Controls – only LB public.
- IAM least privilege – runtime SA: `roles/storage.objectAdmin`, `roles/datastore.user`.
- GDPR – AES-256 at rest, TLS 1.3 in transit; user-delete within minutes.

3.4 DevOps Observability

- CI/CD – GitHub Actions → Artifact Registry → Cloud Run.
- SLIs – p95 latency, error rate; alert if p95 $>$ 1 s or error $>$ 1 %.
- Budget alert at 80 % monthly quota; Cloud Run min-instances = 0.

4 High-Level Design

4.1 Core Abstractions

1. **Image** – immutable original blob (UUID4).
2. **Revision** – result of one atomic op; parent pointer enables undo.
3. **Job** – 1…n operations; sync (10 s) or async.
4. **Pipeline** – ordered op list executed in single Sharp stream.

4.2 Synchronous Flow Diagram

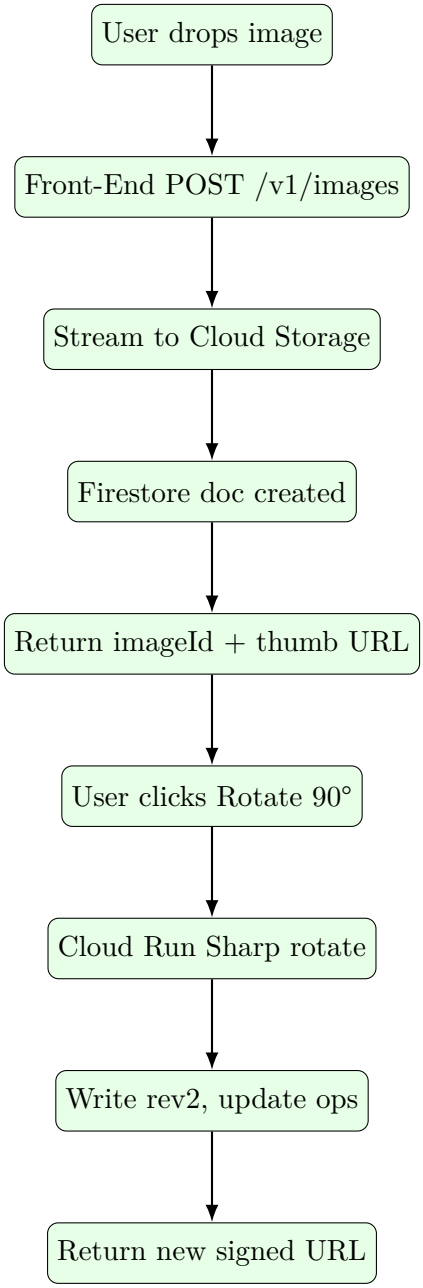

Figure 2: Happy-path single-image transform flow

4.3 Async Batch Flow

1. User selects 20 images → POST `/v1/batch` → returns `jobId`.
2. API pushes 20 tasks to Cloud Tasks.
3. Worker Cloud Run pulls task, downloads, transforms, uploads result.
4. All finish → zip created, Firestore status “done”, SendGrid e-mail with signed URL.

4.4 Scalability Notes

- Sharp (libvips) 8 concurrent transforms/container; Cloud Run max 20 160 concurrent edits.
- Firestore write limit 10 k/s our load.
- CDN cache-hit 85 % keeps egress low.

4.5 Extensibility Hooks

- WASM plugin folder – new filters auto-discovered at cold-start.
- BigQuery daily transfer of ops array → BI dashboard.
- Premium tier – Firebase Auth custom claims gate certain ops.

5 Conclusion

We have presented a complete, cloud-native, serverless design for an Image Editor that meets all functional and non-functional requirements while remaining extensible and cost-efficient for a course project deployment.