Requirements Specification

Owner: Ivan Bakalov

Version: 1.0 (2025-10-05)

Scope: Backend services (Gateway, Auth, Messaging, File). Client apps are out of scope except where needed to define E2EE behavior.

1. Purpose & Goals

Design and operate a secure, privacy-first messaging backend that supports **1:1** and **group** chats and **secure file sharing**. The server must never see plaintext message/file contents (client-side E2EE).

2. Stakeholders & Users

- **End users**: send/receive messages and files, manage groups.
- **Operator**: deploy/operate services, handle support and data requests.
- **Semester coach/Technical teachers**: evaluate learning outcomes, design choices and provide feedback.

3. Definitions

- **E2EE**: Content encrypted/decrypted only on clients; server stores ciphertext & metadata.
- **Session**: Server-side record binding a user to a refresh token, device info, and IP.
- **Conversation**: 1:1 or group channel; has members and message history.
- **Object Storage**: S3/MinIO bucket for encrypted file blobs.

4. System Context (summary)

Clients connect over HTTPS/WS → **Gateway** routes to **Auth**, **Messaging**, and **File** services; **PostgreSQL** stores structured data; **S3/MinIO** stores encrypted files;

optional **Event Bus** for decoupling.

---

5. Functional Requirements

5.1 Identity & Sessions

1. Users can **register** with email + password.
    - Passwords are never logged; server stores only salted hashes.
2. Users can **authenticate**, receive **access** (JWT) + **refresh** tokens.
3. Users can **refresh** tokens; refresh rotates and binds to a single **session**.
4. Users can **logout** (session revoked) from current device.
5. Admin/operator cannot read content; can only view metadata (user id, timestamps, status).

5.2 Public Keys & E2EE Fundamentals

1. Client generates long-term identity key pair locally; server receives **public key** only.
2. For 1:1, the sender derives a shared secret (e.g., X25519) and encrypts message **client-side** with **AES-GCM/ChaCha20-Poly1305**.
3. Server **must not** accept plaintext message payloads; validation rejects such requests.

5.3 Messaging

1. Users can **send/receive** 1:1 messages with the following envelope (server-visible):
    - `message_id` , `conversation_id` , `sender_id` , `created_at` , `ciphertext` , `nonce` , `auth_tag` , `content_type` (e.g., text/file-pointer), optional `parent_id` (reply).
2. **Delivery semantics**: at-least-once; client must de-duplicate via `message_id` .
3. **Status**: store `sent` timestamp; **delivered/read** receipts are optional (if included, they are metadata only).

4. **History**: server persists ciphertext; clients fetch by conversation with pagination.

5.4 Groups

1. Users can **create groups**, **invite** members, and **leave** groups.
2. Group encryption uses a **shared symmetric sender key** per member or per device (rotation on membership change is planned).
3. Server enforces membership for send/receive; it cannot decrypt content.

5.5 File Sharing

1. Users can upload/download **encrypted attachments** up to **50 MB** each (configurable).
2. Client encrypts file before upload; server stores only ciphertext in S3/MinIO and a minimal DB record (owner, size, digest, created_at).
3. Download uses **time-limited pre-signed URLs**; server never streams plaintext.

5.6 Account & Data Subject Rights (GDPR)

1. Users can **export** their account data (profile, metadata, keys, message/file *metadata*, not ciphertext decryption keys).
2. Users can **request deletion** of their account and associated data (see retention policy below).
3. Users can **update** their profile email (with re-verification).

---

6. Non-Functional Requirements (measurable)

6.1 Security

- **Transport**: HTTPS (TLS 1.3 preferred; 1.2 minimum), HSTS enabled on gateway.
- **Passwords**: Argon2id (e.g., time=1, memory≈64 MB, parallelism=2) **or** bcrypt cost ≥ 12.

- **Tokens**: Access TTL **15 min**, Refresh TTL **30 days**; refresh rotation required; session revocation on logout/compromise; 60s clock-skew tolerance.
- **Key handling**: server stores **public keys** only; never accepts or logs private keys.
- **Input validation**: strict JSON schemas; reject plaintext fields where ciphertext is required.
- **Rate limiting**: Auth endpoints **≤ 5 req/sec per IP**, messaging **≤ 60 req/min per user** (configurable).
- **Secrets**: provided via environment (CI/CD); never committed; rotated on demand.

6.2 Privacy, Ethics & GDPR (Privacy-by-Design)

- **Data minimization**: store only what's required (ciphertext, envelopes, minimal metadata). No message content indexing or server-side search.
- **Purpose limitation**: data processed only for messaging; no profiling or secondary use.
- **Lawful basis**: user consent/contract for service provision (portfolio demo context).
- **Data subject rights**: endpoints for **access/export**, **rectification**, **erasure**. Requests verified via email challenge; operator response within **30 days**.
- **Retention**:
  - Accounts/messages/files deleted within **30 days** of confirmed erasure request (soft-delete immediately, purge after grace window).
  - Logs (no content) retained **≤ 30 days** in dev / **≤ 90 days** in prod.
- **Data location**: document storage location (e.g., EU region for S3-compatible store).
- **Security of processing**: encryption in transit & at rest (server stores only ciphertext).
- **Breach handling**: incident log with timestamps, scope, mitigation steps; notify affected users promptly (educational context).

6.3 Availability & Performance

- **Availability target**: ≥ **99.0%** monthly for gateway/auth.
- **Latency (P95)**: Auth endpoints **< 200 ms**; send message **< 500 ms** under baseline load.
- **Throughput**: Support **≥ 50 msgs/sec** and **≥ 100 concurrent WS** connections on dev hardware baseline; document test rig.

6.4 Scalability & Reliability

- Services are stateless and **horizontally scalable** (except DB/object storage).
- Graceful shutdown with in-flight request draining; idempotent processing by `message_id`.
- DB indices on `conversation_id, created_at` for message queries; S3 for blob scale.

6.5 Observability & Operations

- **Structured logs** (no secrets, no plaintext content).
- Health checks in Compose; deployment rollback via immutable `sha-<commit>` image tags.
- Minimal metrics: request count, error rate, P95 latency; log aggregation planned.

6.6 Maintainability & Portability

- Code style enforced via Go linters; ADRs for significant decisions.
- Containerized services; dev/prod parity via Compose and `.env` files.