1. System Overview

A distributed, event-driven microservices platform to manage user profiles and their associated content blocks, with real-time search indexing.

2. Requirements

2.1 Functional Requirements

1. **Profile Management**
    a. Create, read, update, delete user profiles.
    b. Fetch profile by ID or username.
2. **Block Management**
    a. For each profile, maintain an ordered list of "blocks" (text + link).
    b. CRUD operations on blocks.
3. **Event Publishing**
    a. Publish a `ProfileEvent` on create/update/delete profile.
    b. Publish a `BlockEvent` on any change to that profile's blocks.
4. **Search Indexing**
    a. Consume profile and block events.
    b. Keep an Elasticsearch index up to date: documents keyed by username, containing current block list.
5. **Service Discovery & Routing**
    a. All services register with Eureka.
    b. External APIs routed via API Gateway.

2.2 Non-Functional Requirements

1. **Scalability**: Each microservice and Kafka/Elasticsearch clusters must scale independently.
2. **Reliability**:
    a. Kafka topics must guarantee at-least-once delivery.
    b. Consumers handle idempotent writes to Elasticsearch.
3. **Performance**:
    a. Profile and block APIs respond within 200 ms under normal load.

   b. Search queries to Elasticsearch return results in under 100 ms for 1 million documents.
4. **Maintainability**:
   a. Code organized into clear service boundaries.
   b. Well-documented APIs and events.
5. **Observability**:
   a. Each service emits structured logs.
   b. Metrics on event publishing/consumption latencies.

ID	Name	Actor	Precondition	Flow	Postcondition
UC1	Create Profile	Client App	User authenticated	1. Client → API-Gateway → Profile Service POST `/profiles` (username)	

2. Service generates ID, saves to PostgreSQL
3. Service publishes `ProfileEvent` to Kafka
4. Service calls Block Service to init empty blocks | Profile stored; event emitted; empty block document created |
   | UC2 | Update Blocks | Client App | Profile exists; user authorized| 1. Client → API-Gateway → Block Service PUT `/profiles/{id}/blocks` (block list)
5. Service updates MongoDB
6. Service publishes `BlockEvent` to Kafka | Blocks updated; event emitted |
   | UC3 | Search Profiles by Term | Client App | Elasticsearch index up to date | 1. Client → API-Gateway → Search API
7. Query ES index for matching usernames or block content
8. Return paginated results | Results include matching profiles + blocks |
   | UC4 | Full Reindex Profile | Admin Tool | Profile exists | 1. Admin tool calls Search-Aggregator Service `/reindex/{profileId}`
9. Service fetches profile & blocks
10. Service writes full document to ES | ES document overwritten with latest data |

4. Domain Objects & Classes

Object / Class	Responsibilities	Relationships
Profile	`id`, `username`	1–1 with Block Collection; emits ProfileEvent
Block	`type`, `text`, `link`	Nested inside Content / ES document

BlockEvent	`profileId`, `timestamp`	Kafka message
ProfileEvent	`profileId`, `timestamp`	Kafka message
ProfileService	CRUD on Profile; publishes ProfileEvent; calls Block Service	uses ProfileRepository, KafkaTemplate
BlockService	CRUD on blocks; publishes BlockEvent	uses BlockRepository, KafkaTemplate
SearchIndexAggregatorService	Consumes events, calls ProfileClient & BlockClient, updates ES	uses ElasticsearchClient or ES Repo
ProfileSearchDocument	ES document: `username`, `List<Block>`	stored in `profiles` index
ProfileRepository	Spring Data JPA for Profile → PostgreSQL	
BlockRepository	Spring Data MongoDB for Content → MongoDB	
ProfileClient / BlockClient	Feign client interfaces to fetch remote data	
API Gateway	Routes external HTTP calls to microservices	
Eureka Client / Server	Service discovery	

5. CRC Cards

Class	**Responsibilities**	**Collaborators**
ProfileService	– Manage profiles– Publish ProfileEvent	ProfileRepository, KafkaTemplate, BlockClient
BlockService	– Manage blocks– Publish BlockEvent	BlockRepository, KafkaTemplate
SearchIndexAggregator	– Listen to Kafka topics– Update ES index	ProfileClient, BlockClient, ProfileSearchRepo