**GROWNET — System Architecture Document**

---

## 1. Purpose & Principles

**Purpose**

»طراحی معماری‌ای که **قابل رشد**، **امن**، **قابل مشاهده** و **قابل تغییر کنترل‌شده** باشد؛ نه سیستمی که با رشد کاربر بشکند«.

**Architectural Principles**

- **Scalability by design**
- **Loose coupling, strong contracts**
- **Data as a first-class citizen**
- **Security by default**
- **Cost-aware architecture**

---

## 2. System Overview (High-Level)

GROWNET یک **platform-centric system** با ویژگی‌های:

- user-generated content
- ranking & reputation engine
- monetization
- B2B analytics

**Architecture Style**

- **Service-Oriented / Modular Monolith → Microservices-ready**
- Event-driven where needed

دلیل:
در early-stage، microservice کامل = هزینه و پیچیدگی زودرس.

---

## 3. High-Level Architecture Diagram (Conceptual)

```
[ Web / Mobile Client ]
|
[ API Gateway ]
|
---------------------------------------------
| Auth | User | Content | Ranking | Monetization |
----------------------------------------------
                                             |
                                 [ Data Layer ]
                                             |
                      [ Observability & Infra ]
```

---

## 4. Core Services & Boundaries

### 4.1 Auth & Identity Service

- Authentication (OAuth / Email)
- Authorization (RBAC)
- Token management (JWT)

**Why separated?**
Security isolation + legal compliance

---

### 4.2 User & Profile Service

- User metadata
- Credibility profile
- Roles (individual / company / admin)

---

### 4.3 Content Service

- Content CRUD
- Tagging & categorization
- Moderation flags

---

### 4.4 Ranking & Reputation Engine

- Signal aggregation
- Score calculation
- Anti-gaming logic

**Most critical service**

---

### 4.5 Monetization Service

- Credits / Stars
- Earnings
- Fraud detection hooks

---

#### 4.6 Company & Analytics Service

- Company pages
- Feedback aggregation
- Dashboard metrics

---

## 5. Data Flow (Key Scenarios)

#### Scenario A — Content Publish

1. User authenticated
2. Content Service stores content
3. Event emitted: content_created
4. Ranking Service listens → recalculates score

---

#### Scenario B — Monetized Interaction

1. User reacts/comments
2. Monetization Service validates
3. Credits transferred
4. Ranking updated

---

## 6. Data Architecture

#### Databases

| Service | DB | Reason |
|---|---|---|
| User / Auth | PostgreSQL | ACID, relations |
| Content | PostgreSQL | Structured content |
| Ranking | Redis + Postgres | Speed + persistence |
| Analytics | ClickHouse / BigQuery | Aggregation at scale |
| Logs | Elastic / Loki | Observability |

**Polyglot persistence by need**

---

## 7. API Design

**API Style**

- REST (early)
- GraphQL (read-heavy dashboards later)

**Example Endpoint**

POST /api/v1/content

GET  /api/v1/profile/{id}

**Contract Rules**

- Versioned APIs
- Backward compatibility
- OpenAPI specs

---

## 8. Security Architecture (AuthN/AuthZ)

**Authentication**

- OAuth2
- JWT tokens
- Refresh token rotation

**Authorization**

- RBAC:
  - User
  - Company Admin
  - System Admin

**Security Controls**

- Rate limiting
- Anti-fraud scoring
- Content abuse detection

---

## 9. Privacy & Compliance

- GDPR-ready

- Right-to-forget
- Data minimization
- Audit logs

---

**10. Observability**

**Metrics**

- Request latency
- Error rates
- User actions

**Logs**

- Structured logs
- Correlation IDs

**Tracing**

- Distributed tracing (OpenTelemetry)

بدونobservability ، رشد = نابینایی

---

**11. Infrastructure**

**Cloud Strategy**

- AWS / GCP
- Container-based (Docker)
- Kubernetes (later stage)

**CI/CD**

- Automated tests
- Canary releases
- Rollback strategy

---

**12. Scalability Strategy**

| Layer | Strategy |
|---|---|
| API | Horizontal scaling |

| Layer | Strategy |
|---|---|
| DB | Read replicas |
| Ranking | Caching + async |
| Analytics | Batch processing |

Ranking decoupled from user-facing latency

---

## 13. Cost Control

- Autoscaling
- Tiered storage
- Avoid over-engineering

معماری خوب = هزینه آینده قابل پیش‌بینی

---

## 14. Failure Modes & Resilience

| Failure | Mitigation |
|---|---|
| DB overload | Read replicas |
| Ranking delay | Async queue |
| Service crash | Circuit breaker |
| Fraud attack | Throttling |

---

## 15. Architecture Decision Records (ADR)

### ADR-001 — Modular Monolith

**Decision:** Avoid early microservices
**Reason:** Team size & speed
**Tradeoff:** Refactor later

---

### ADR-002 — PostgreSQL as Core DB

**Decision:** Single strong relational DB
**Reason:** Consistency & simplicity

---

### ADR-003 — Async Ranking

**Decision:** Event-driven ranking
**Reason:** Performance & isolation

---

### 16. Data Consistency Model

- Strong consistency for payments
- Eventual consistency for ranking

---

### 17. Dependency Map

| Team | Dependency |
|---|---|
| Legal | Monetization rules |
| Sales | Pricing logic |
| Data | Analytics schemas |

---

### 18. Security Risks & Mitigation

| Risk | Mitigation |
|---|---|
| Credential leak | Token rotation |
| Abuse | Behavior analysis |
| Injection | Input validation |

---

### 19. Evolution Roadmap (Architecture)

| Phase | Change |
|---|---|
| MVP | Modular monolith |
| Scale | Split ranking & analytics |
| Growth | Dedicated fraud service |

---

### 20. What We Are NOT Doing (Discipline)

- No blockchain
- No premature ML
- No full microservices

این بخش نشان بلوغ است.

---

## 21. Architecture Success Criteria

- 10x user growth without rewrite
- Predictable infra cost
- Clear ownership per service

---

## 22. Investor / CTO Takeaway

«این معماری نشان می‌دهد تیم می‌داند کجا ساده بماند و کجا پیچیدگی را آگاهانه اضافه کند.»

---

## 23–30. (PDF Pages)

در نسخه PDF نهایی:

- دیاگرام‌های حرفه‌ای
- Sequence diagram
- Data flow diagram
- Threat model
- ADR appendix