Great — I'll create **all three deliverables** you asked for:

1️⃣ **6-month SDE2 → SDE3 project timeline (Mermaid Gantt)**
2️⃣ **10-project architecture diagram set (Mermaid)**
3️⃣ **Standard GitHub repo folder structure for all 10 projects**

Everything written so you can copy-paste directly into GitHub/Notion.

✅1️⃣ 6-MONTH SDE2 → SDE3 PROJECT TIMELINE (MERMAID GANTT)

Paste this into any Mermaid-supported renderer:

```
gantt
    title SDE2 → SDE3 Six-Month Project Timeline
    dateFormat  YYYY-MM-DD
    section Month 1: Core Distributed Systems
    Distributed Rate Limiter (Go + Redis)          :a1, 2025-01-01,
20d
    Real-Time Chat & Presence Service              :a2, 2025-01-10,
20d

    section Month 2: Event Systems & Reliability
    Notification Delivery Platform (Kafka)         :b1, 2025-02-01,
25d
    Distributed Cron Scheduler (Etcd + gRPC)       :b2, 2025-02-10,
25d

    section Month 3: High-Scale Architecture
    Event-Driven E-Commerce System                 :c1, 2025-03-01,
30d
    API Gateway (Routing + Auth + Rate Limit)      :c2, 2025-03-15,
20d

    section Month 4: Data & ML Systems
    ETL Data Pipeline (Airflow + Python)           :d1, 2025-04-01,
25d
    Recommender System Backend (FAISS + Python)    :d2, 2025-04-10,
20d
```

```
    section Month 5: Advanced Infrastructure
    Multi-Region Deployment (AWS + Terraform)      :e1, 2025-05-01,
30d
    Real-Time Analytics Engine (Kafka + ClickHouse):e2, 2025-05-15,
25d

    section Month 6: Polish, Docs, and Promo
    Documentation + Architecture Diagrams          :f1, 2025-06-01,
15d
    Resume + Portfolio + LinkedIn Upgrade          :f2, 2025-06-15,
10d
    Apply for SDE3/Remote Roles                    :f3, 2025-06-20,
10d
```

This 6-month plan mirrors an *actual promotion roadmap* inside product companies.

✅2️⃣ ARCHITECTURE DIAGRAM SET FOR ALL 10 PROJECTS (MERMAID)

Each diagram shows **core system architecture** — ready for GitHub READMEs.

---

1. Event-Driven E-Commerce System

```
flowchart LR
    User --> APIGW
    APIGW --> Auth
    APIGW --> Catalog
    APIGW --> Cart
    APIGW --> Order
    Order --> Payment
    Order --> Kafka((Kafka))
    Kafka --> Inventory
    Kafka --> Notification
    Inventory --> DB[(PostgreSQL)]
    Catalog --> Cache[(Redis)]
```

---

2. Real-Time Ride Matching System (Uber Mini)

```
flowchart LR
    DriverApp --> WS((WebSocket))
    RiderApp --> WS
    WS --> MatchService
    MatchService --> GeoIndex[(Redis Geo)]
    MatchService --> Kafka((Kafka))
    Kafka --> TripService
    TripService --> DB[(PostgreSQL)]
    MatchService --> ETAEngine
```

---

3. Distributed Rate Limiter

```
flowchart LR
    Client --> SDK
    SDK --> GRPC
    GRPC --> LimiterService
    LimiterService --> Redis[(Redis Lua Scripts)]
    LimiterService --> Metrics((Prometheus))
```

---

4. Notification Delivery Platform

```
flowchart LR
    Backend --> Kafka((Kafka))
    Kafka --> WorkerPool
    WorkerPool --> Email
    WorkerPool --> SMS
    WorkerPool --> Push
    WorkerPool --> DLQ((Dead Letter Queue))
    WorkerPool --> Logs[(ElasticSearch)]
```

---

5. Feature Flag System

```
flowchart LR
    AdminUI --> FlagService
    FlagService --> DB[(PostgreSQL)]
    BackendApps --> SDK
    SDK --> FlagService
    FlagService --> Cache[(Redis)]
```

---

6. Real-Time Chat + Presence Service

```
flowchart LR
    UserA --> WS((WebSocket))
    UserB --> WS
    WS --> ChatService
    ChatService --> RedisStreams
    ChatService --> PresenceService
    PresenceService --> Redis[(Online/Offline)]
    ChatService --> DB[(Messages DB)]
```

---

7. Distributed Cron Scheduler

```
flowchart LR
    AdminUI --> Scheduler
    Scheduler --> Etcd[(Leader Election)]
    Scheduler --> Workers
    Workers --> TaskQueue((Queue))
    Workers --> Logs[(ElasticSearch)]
```

---

8. API Gateway

```
flowchart LR
    Client --> APIGateway
    APIGateway --> AuthService
    APIGateway --> RateLimiter
    APIGateway --> Router
    Router --> MicroservicesA
    Router --> MicroservicesB
```

```
    RateLimiter --> Redis
```

---

9. ETL Data Pipeline

```
flowchart LR
    SourceAPI --> Extract
    Extract --> Transform(Python + Pandas)
    Transform --> Load
    Load --> Warehouse[(PostgreSQL)]
    Load --> S3[(S3 Data Lake)]
    Orchestrator(Airflow) --> Extract
    Orchestrator --> Transform
    Orchestrator --> Load
```

---

10. Recommender System Backend

```
flowchart LR
    User --> FastAPI
    FastAPI --> EmbeddingModel
    EmbeddingModel --> VectorIndex((FAISS))
    FastAPI --> DB[(User Data)]
    Scheduler --> BatchIndexer
    BatchIndexer --> VectorIndex
```

3️⃣ STANDARD GITHUB REPO STRUCTURE FOR ALL 10 PROJECTS

You can reuse this for every project — it looks professional and SDE2/SDE3-level.

```
project-name/
│
├── cmd/
│   └── service-name/       # Main entry points
│
├── internal/
│   ├── handlers/           # HTTP/gRPC handlers
│   ├── services/           # Business logic
│   ├── repository/         # DB access logic
│   ├── queue/              # Kafka/RabbitMQ consumers
│   ├── cache/              # Redis caching layer
│   ├── config/             # Config loading
│   └── middleware/         # Auth, rate-limit, logging
│
├── pkg/                    # Utilities shared across modules
│
├── api/
│   ├── proto/              # gRPC definitions
│   └── openapi/            # Swagger/OpenAPI specs
│
├── deployments/
│   ├── docker/             # Dockerfiles
│   ├── k8s/                # Kubernetes manifests
│   └── terraform/          # IaC configs
│
├── scripts/                # Bash/Python utility scripts
│
├── docs/
│   ├── architecture.md
│   ├── sequence-diagrams/
│   └── system-design.md
│
├── tests/                  # Unit + integration tests
│
├── Makefile
├── docker-compose.yml
```

```
├── README.md
└── go.mod / requirements.txt / package.json
```

This repo structure **looks like real production engineering**, not student work.
