SDF2 — Visual Programming Lesson: System Design Fundamentals (Build Track)

Audience: 7 groups × 4 core roles (**FRONT**, **BACK**, **DATABASE**, **SMART CONTRACTS**)
Mode: Visual-first orchestration with thin code where needed (flows, diagrams, and dashboards drive the build)

> SDF2 **extends SDF1's modeling work (UML + ERD)** into a working MVP, and reuses the **visual-flow DApp approach** (click contract event ETL UI/alert) from the DApp lesson.

1) Purpose & Outcomes

By the end of SDF2, each group ships a **demoable MVP** that:

- Implements **end-to-end user stories** with **visual workflows** that stitch together Front, Back, DB, and Smart Contracts.
- Preserves **traceability** back to SDF1's **UML/ERD** models (class/sequence/activity diagrams endpoints, tables, and events).
- Exposes a **metrics dashboard** and a **short report** (what was built, known trade-offs, and measured results).

2) Project Scope

Pick **one small, demoable scope** (e.g., NFT ticketing, milestone crowdfunding, or a simple marketplace). Keep features minimal and polish deep (TX lifecycle UX, event-driven updates, indexed history).

Required user stories (customize to your scope):

1. As a user, I can perform the **primary action** so that I get **immediate feedback** and can see **history**.
2. As a user, I see **real-time status** after my transaction so I know it **finalized**.
3. As an admin, I can **resolve/approve/retire** with an **audit trail**.

For each story include: **Given/When/Then**, **visual-flow screenshot**, **expected events & DB rows**.

3) Architecture & Tech Baseline

Visual Orchestration: Node-RED / n8n for the runnable flow (triggers, contract calls, event listeners, ETL, alerts).

FRONT

- **Stack:** Next.js (SSR + CSR), HTML5, Tailwind CSS (minimalistic color grading, smooth scrolling), Three.js (user interacting animations).
- **UI/UX requirements:** Color theory basics; **SSR + CSR** strategy; **SEO** (meta, sitemap, OG/Twitter); **lazy loading** (dynamic imports); **skeleton loading**; components: **Carousel, Popup/Modal, Toast/Snack Bar, Accordion, Bento Grid, Bread-Crumbs, Checklist, Button, Radio Button, Tabs**; accessibility (labels, tab order, contrast).
- **TX lifecycle UX:** pending success/fail; event-driven refresh from indexer.

BACK

- **Web framework:** Python **Django (MVT)** or **Flask (MVC/MVVM)**.
- **API layer: FastAPI** (REST) **and GraphQL** schema; provide a simple **RPC** (JSON-RPC or gRPC stub) for the indexer/desktop GUI.
- **Concurrency & sync:** ASGI (uvicorn) where needed; background jobs (Celery/cron) for batch ops; **asynchronous I/O synchronization** for event ingest DB.
- **Edge: Load balancer** design (e.g., Nginx/HAProxy) and rate limiting on public endpoints.

GUI (Desktop, optional but recommended for Admin/Operator)

- **PySide6 / PyQt6** desktop panel for back-office actions (admin resolve, manual re-index, health view) via WebSocket/HTTP APIs.

DATABASE

- Choose **one primary OLTP** (Postgres/MySQL/OracleSQL/MsSQL) and optionally **DuckDB** for analytical views.
- **Data design:** Normalized schema (≤4NF) aligned with SDF1 ERD; **migrations** + **seed**; **indexes**, **query optimization**, **caching**, **batch operations** (bulk inserts for event backfills); **one materialized view** for dashboard.

SMART CONTRACTS

- **Solidity** on a testnet; emit events for UI/ETL needs.
- **Brownie** for deployment/testing; **≥10 unit tests**, including at least one failure path; publish test report + deployed address.

Observability, Security, & Docs

- **Metrics dashboard:** at least 4 KPIs (TX success rate, median event lag, primary conversion %, and an internal performance metric), plus **one alert** (e.g., event lag > 60s, error rate > 2%).
- **Threat model one-pager:** top 3 risks + mitigations; input schema validation; rate limiting on admin paths.
- **Docs:** 1-page README with architecture sketch and story mapping to flows.

4) Role-by-Role Tasks (FRONT / BACK / DATABASE / SMART CONTRACTS)

FRONT

- Implement SSR/CSR split, SEO metadata, lazy-loaded islands, skeleton loading, and the required UI kit (Carousel, Modal/Popup, Toast/Snack Bar, Accordion, Bento Grid, Breadcrumbs, Checklist, Button, Radio Button, Tabs), User Interacting Animations.
- Wallet connect + TX status toasts; subscribe to **indexed events** for real-time updates; export a 60-sec demo clip.

BACK

- Indexer: listen decode persist emit (WS/HTTP) with **asynchronous I/O**; expose REST/GraphQL; one **batch worker** for backfills.
- Add **rate limiting** and **auth** on admin endpoints; produce OpenAPI + GraphQL schema.

DATABASE

- Deliver ERD, migrations, seed data; prove **normalization** (≤4NF) and **query plans** with chosen indexes; add one **materialized view** for the dashboard.

SMART CONTRACTS

- Implement minimal contract; emit events for state changes; **Brownie** test suite (≥10 tests, coverage on core functions), deploy to testnet, and document addresses & ABIs.

5) MVP Requirements (pass/fail gates)

- **Visual workflow** runs end-to-end with: Trigger Contract Call Event Listener/Indexer ETL DB/API Decision/Alert UI update.
- **Contract** on testnet + tests/coverage report.
- **Frontend** renders SSR initial view; uses CSR hydration; shows TX lifecycle & event-driven updates.
- **DB** seeded; migrations reproducible; one materialized view; query plan screenshots for 2 hot paths.
- **Monitoring** dashboard with 4 KPIs + 1 alert; threat model note.

6) Deliverables (one package per group)

1. **README (1 page)** — problem, scope, user stories, architecture sketch, SSR/CSR plan.
2. **Visual Flow** — exported PNG + flow definition (e.g., `.json`).
3. **Contracts** — Solidity source, Brownie tests + coverage, deployed address.
4. **API Spec** — OpenAPI + GraphQL schema.
5. **DB** — ERD PNG, migrations, seed, materialized view DDL.

6. **Monitoring & Security** — threat model, dashboard screenshots, alert config.
7. **Front UX bundle** — component checklist and Lighthouse report.

7) Evaluation Criteria (10 pts total)

A. UI/UX & Frontend Engineering — 2 pts

- **Color Theory & Accessibility** (contrast ≥ AA; consistent palette): **0.5**
- **SSR + CSR** (isomorphic routing, hydration without flicker; **skeletal loading** in critical views): **0.5**
- **SEO** (meta, sitemap, robots, OG/Twitter, Lighthouse SEO ≥ 90): **0.5**
- **Performance** (lazy loading via dynamic imports; p95 TTI ≤ 3.5s on demo data): **0.5**
- **Component Set** implemented and accessible: **Carousel, Popup, Toast/Snack Bar, Accordion, Bento Grid, Bread-Crumbs, Checklist, Button, Radio Button, Tabs** (documented with screenshots): **0.5**

B. Backend, API & Systems — 2 pts

- **Framework quality** (Django/Flask structure with clear MVC/MVT; typed FastAPI endpoints; GraphQL schema with 2 non-trivial resolvers): **0.5**
- **RPC** endpoint (JSON-RPC or gRPC) used by indexer/GUI with one streaming or bidirectional call: **0.5**
- **Asynchronous I/O synchronization** (non-blocking event ingest, idempotent consumers, at-least-once handling): **0.5**
- **Load Balancer & Ops** (diagram + working config; per-IP rate limiting on public endpoints): **0.5**
- **Security** (input validation, auth on admin endpoints, minimal secrets hygiene): **0.5**

C. Database Design & Performance — 2 pts

- **Normalization & Consistency with SDF1 ERD** (≤4NF, FK integrity, aligns with modeled entities): **0.5**
- **Indexing & Query Optimization** (show `EXPLAIN`/plans; reduce two hot queries to p95 ≤ 150 ms): **0.5**
- **Caching** (read-through or HTTP caching with hit-rate ≥ 60% on a hot endpoint): **0.5**
- **Batch Operations** (bulk insert/upsert for event backfills with progress logging): **0.5**

D. Smart Contracts — 2 pts

- **Correctness & Events** (clean state transitions; events for each UI-relevant change): 0.5
- **Testing (Brownie)** ≥ **10 tests**, include failure paths; **coverage ≥ 80%** on core functions: **0.5**
- **Deployment & Docs** (testnet address, ABI, minimal README for integration): **1**

E. Visual Flow, Monitoring & Demo — 2 pts

- **Visual Workflow Quality** (deterministic paths, error branches, retries): **0.5**
- **KPIs & Alert** (TX success rate, median event lag, conversion %, plus one internal metric; working alert when lag > 60s or error rate > 2%): **0.5**
- **Demo & Documentation** (≤3 min, crisp narration, artifacts reproducible): **1**

Pass/Fail gates: runnable visual flow, testnet contract with tests, SSR+CSR frontend with TX lifecycle, seeded DB + materialized view, dashboard + one alert.

8) Suggested Milestones

- **Sprint 0 (Setup):** repo, CI, environments, baseline ERD schema, Next.js shell with Tailwind, contract scaffold.
- **Sprint 1 (Contract & Indexer):** Solidity core + Brownie tests; event indexer writing to DB; visual flow skeleton.
- **Sprint 2 (Front & APIs):** SSR pages + CSR islands, UI kit, FastAPI/GraphQL live, TX lifecycle, SEO pass.
- **Sprint 3 (Perf & Ops):** indexing performance, caching, batch backfills, load balancer, rate limiting, dashboard & alert.
- **Sprint 4 (Polish):** QA, docs, demo video, finalize rubric artifacts.

Submission Checklist

- `docs/` README, SDF1 SDF2 traceability note (diagrams implementation), threat model, Lighthouse & plans
- `flows/` visual-flow PNG + importable JSON
- `contracts/` Solidity + Brownie tests + coverage + addresses
- `backend/` Django/Flask + FastAPI/GraphQL specs, RPC stub, `.env.example`
- `db/` ERD PNG, migrations, seed, materialized view DDL, query plans
- `frontend/` component screenshots & accessibility notes
- `monitoring/` dashboard screenshots + alert config
- `demo/` ≤3-minute video