JSCOPE Presents...

Fund Good Deeds

JSCOPE Team: Connor Bashaw, Oliver Gomes, Jonathan Ho, Patrick Lebeau

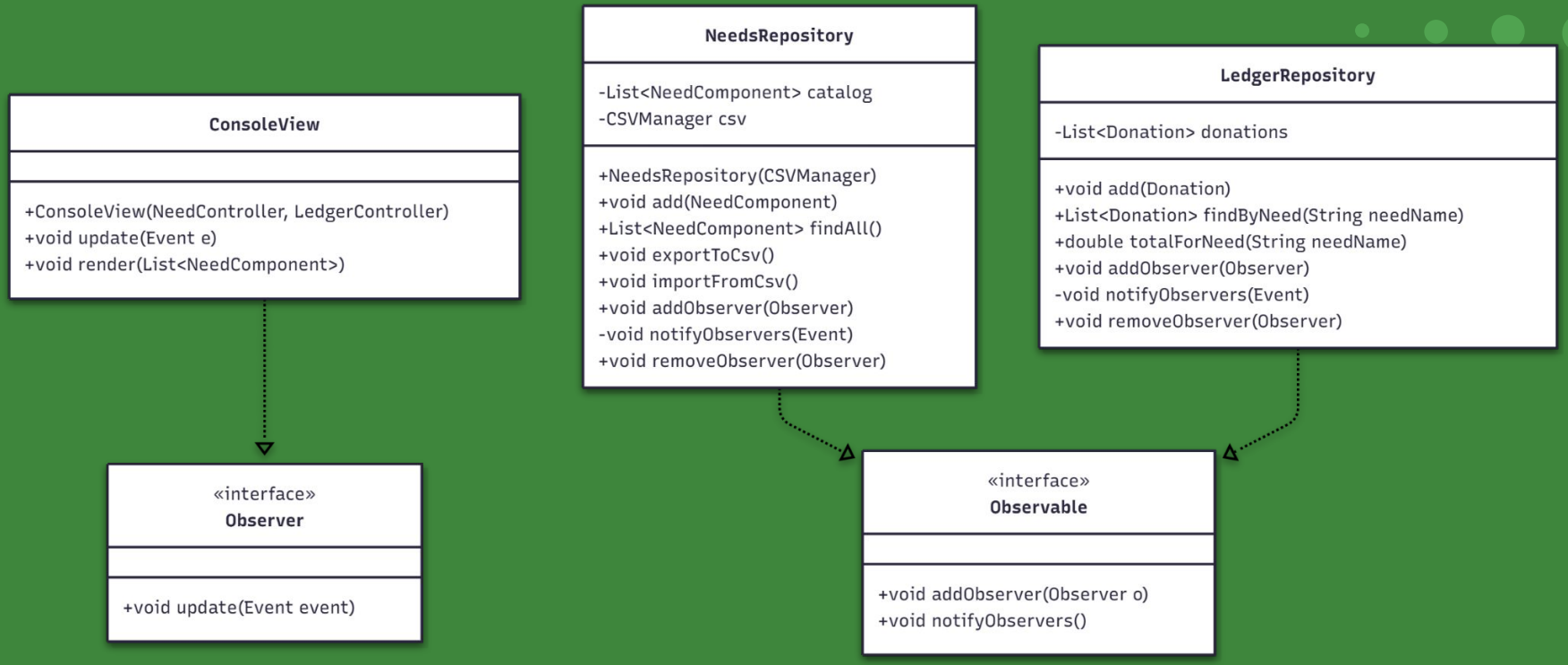
PROBLEM & GOALS

Problem: Donors want transparent, immediate feedback that their contribution pushes a **specific Need** to fulfillment.

V1.0 Goals (today):

- Create a **Need** (description, cost)
- **Donate** to a Need (recorded in **Ledger**)
- **Observer** updates views automatically
- **Fulfilled** state when fundedAmount ≥ cost

Observer Pattern



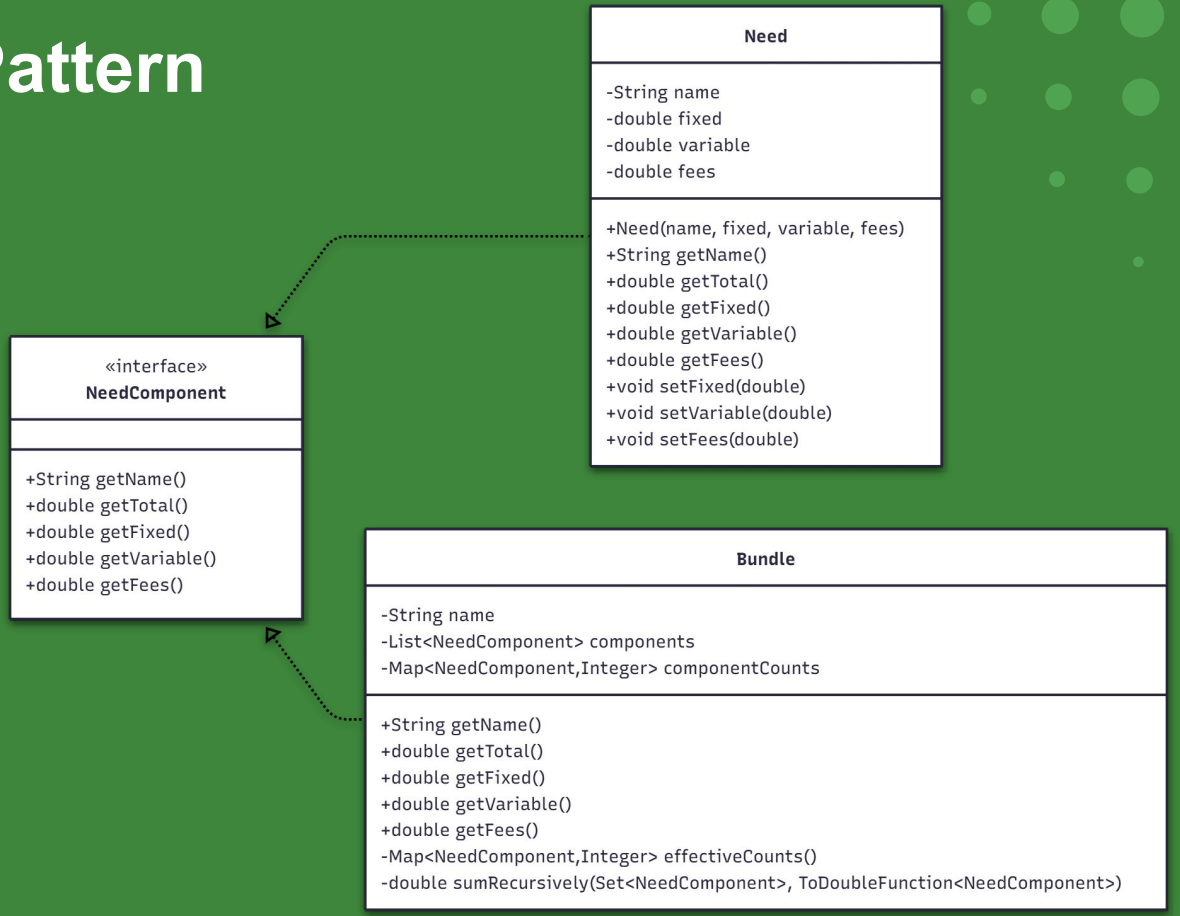
Composite Pattern

Core Functionality:

Shared interface Bundle(s) in Bundle capabilities

Important recursive functions:

getName()
getTotal()



Core Principles

Strengths:

Low Coupling
Separation of Concerns

Dependency Inversion Principle:

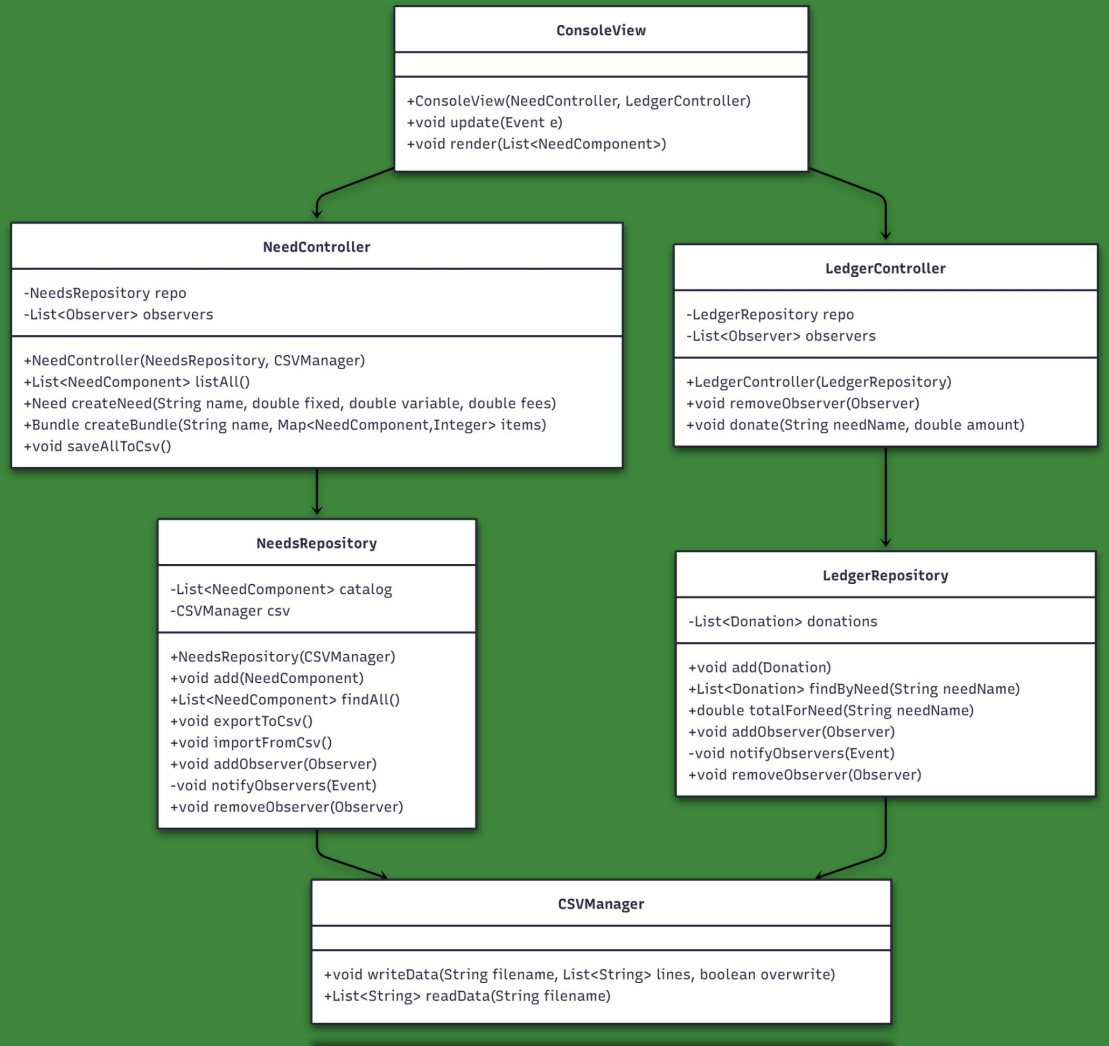
Via Dependency Injection

MVC Architecture

VIEW
- Console/CLI

CONTROLLER
- NeedsContr.
- LedgerContr.

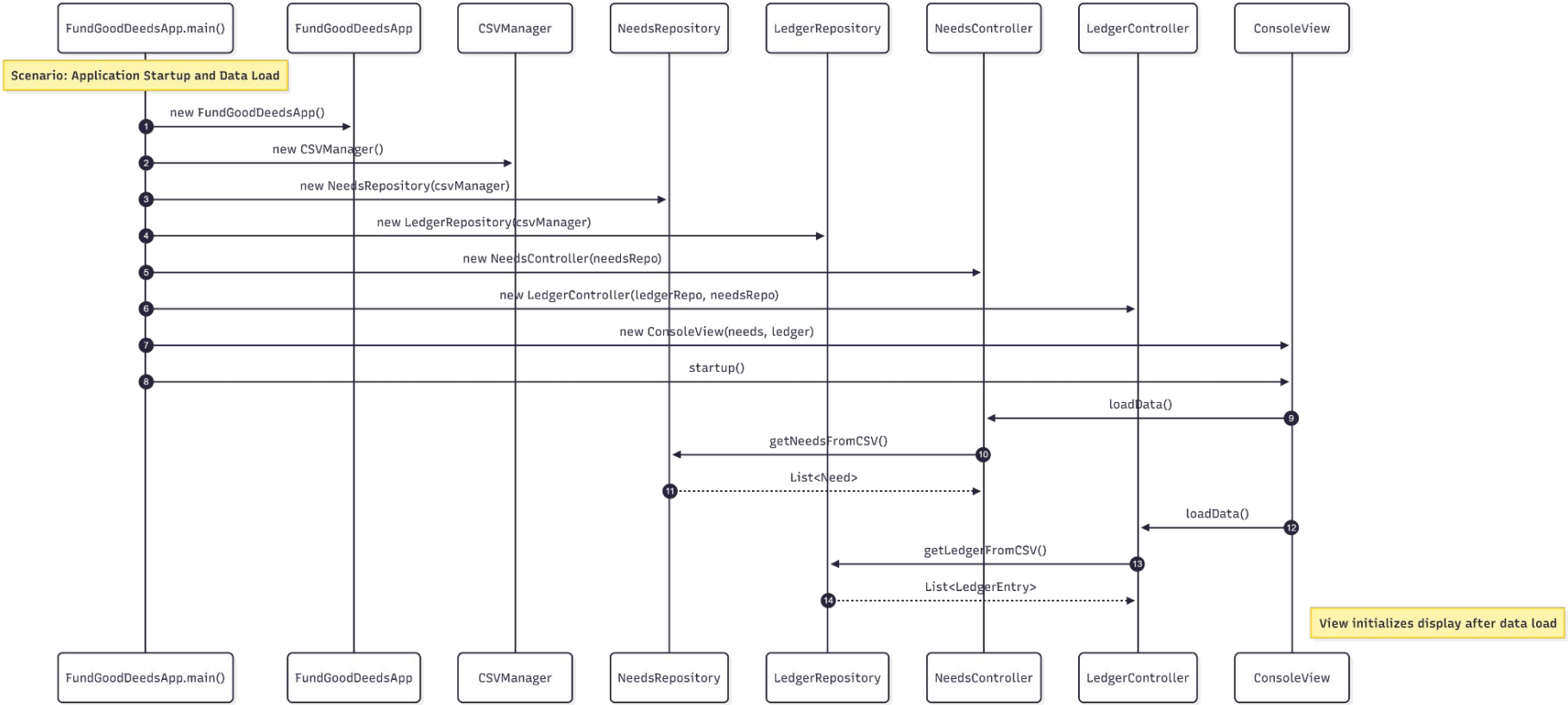
MODEL
- Repos
- CSV Manager



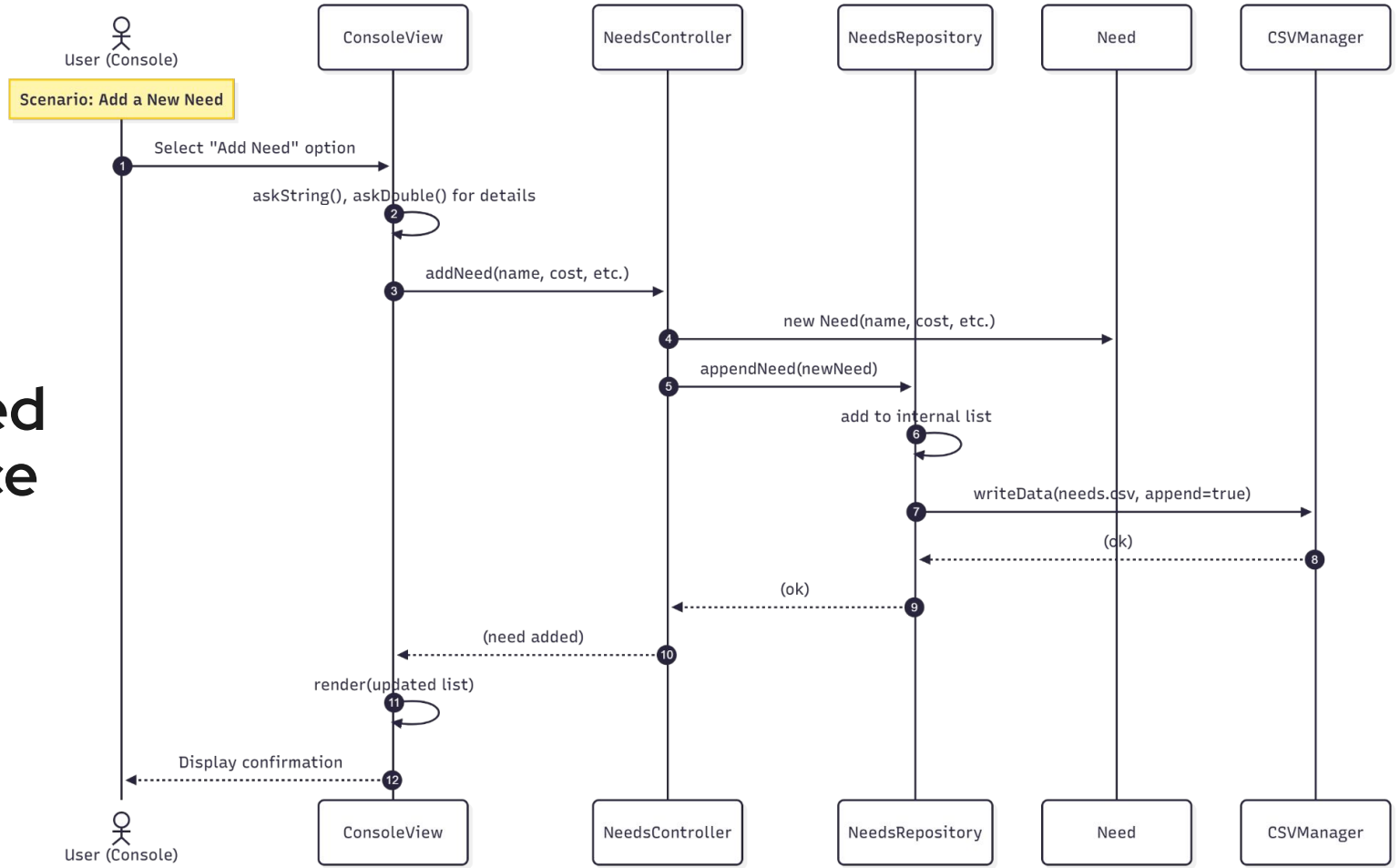
Decisions and tradeoffs

- **Observer over polling** → Event-driven, lower coupling, instant updates
- **CSV/in-mem now** → Fast iteration, deterministic demos, simple seeding for testing.
- **Interface-first Repos** → Storage swap with no domain/controller changes
- **Central Ledger** → One source of truth for donations/transactions
- **Drawbacks** → Low cohesion for model components

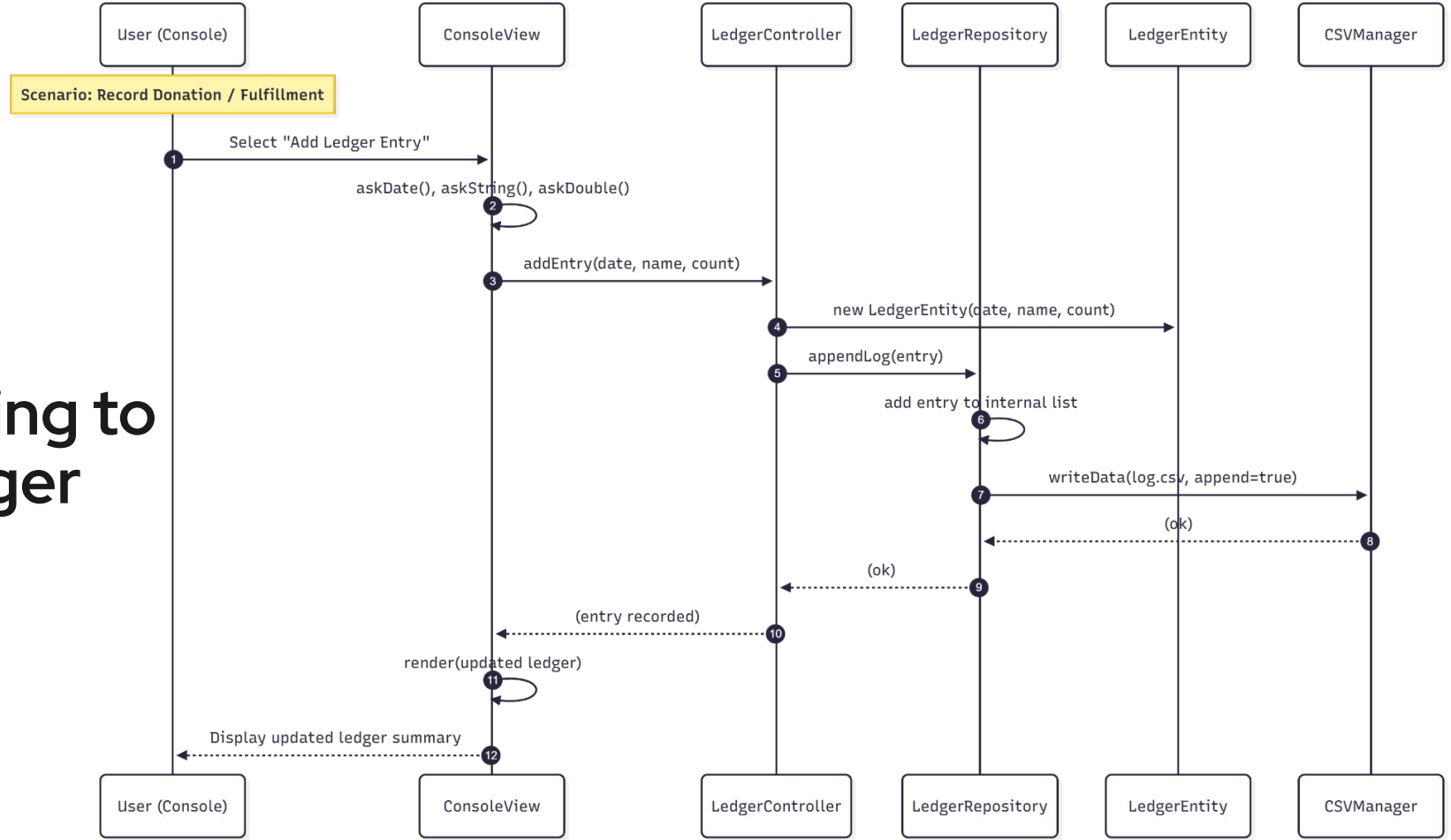
Startup Sequence



New Need Sequence



Adding to Ledger



Live Demo

This demonstration showcases the Composite (Needs/Bundles) and Observer (Model updates View) patterns as the core of the application.

Key Demonstration Points:

1. **Create a New Need (leaf)** → Add a non-composite item to the catalog.
2. **Record Fullfillment** → Add a donation/fulfillment entry to the Ledger.
3. **Observer in Action** → Observe the instant **[ALERT]** messages that confirm the View is automatically updated when the **Model** changes.
4. **Create a Bundle (Composite)** → (if time allows) Assemble existing Needs into a new Bundle

Team Roles and challenges

Technical Roles:

- **Connor:** View & Integration
- **Jon:** Controllers & Business Logic (create/donate flows, wire repos ↔ model, Quality Assurance)
- **Patrick:** Persistence & Repository (CSV read/write, storage logic, seeds)
- **Oliver:** Domain & Composite (Need/Bundle design, Composite behavior, Quality Assurance)

Process: GitHub issues → PRs → reviews, one-command run.

Challenges → Mitigation: controller timing → mocks, seed consistency → CSV versioning, demo risk → backups + projector-safe fonts.

Next Steps & Q&A

Next

- Visual UI probably SwingUI
- Validation & error paths
- Basic auth/roles
- Better testing for edge cases
- Bundle allocation strategies
- Donation fulfillments are reflected in our Needs Catalog