

A decorative pattern of green squares and rectangles of varying shades, arranged in a grid-like fashion that tapers off towards the right side of the image. The pattern is set against a dark grey background.

JSCOPE Presents...

Fund Good Deeds

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

Team

Oliver – Model Lead

- Need , Funding, Ledger
Entities and business
Rules

Jonathan – Controller Lead

- Need Controller, Funding Controller,
Master Controller

Patrick – Persistence Lead

- CSV Load / Save
- Repositories
- Data integrity

Connor – View Lead + PM

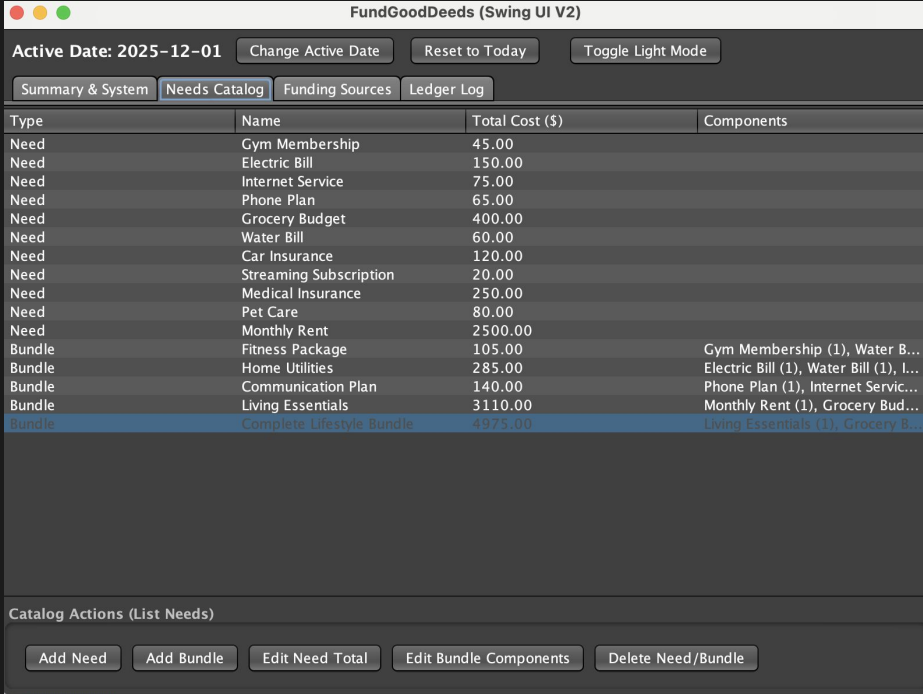
- Swing UI, CLI, story breakdowns,
pacing.

What We're Solving

- Expand Fund Good Deeds App with funding sub system
- Implement new repos, and Master Controller to coordinate workflow
- Add GUI Functionality
- Update Architecture diagrams
- Maintain MVC Compliance
- Follow V2 Requirements from site.

R2 Solution

- Funding Source Implementation
 - Funding Controller, and Funding Repository
- Master Controller for cross subsystem actions
- Need and Funding allocation actions
- Swing UI Implementation
- Observer Based UI Refresh
- Update class + sequence Diagrams

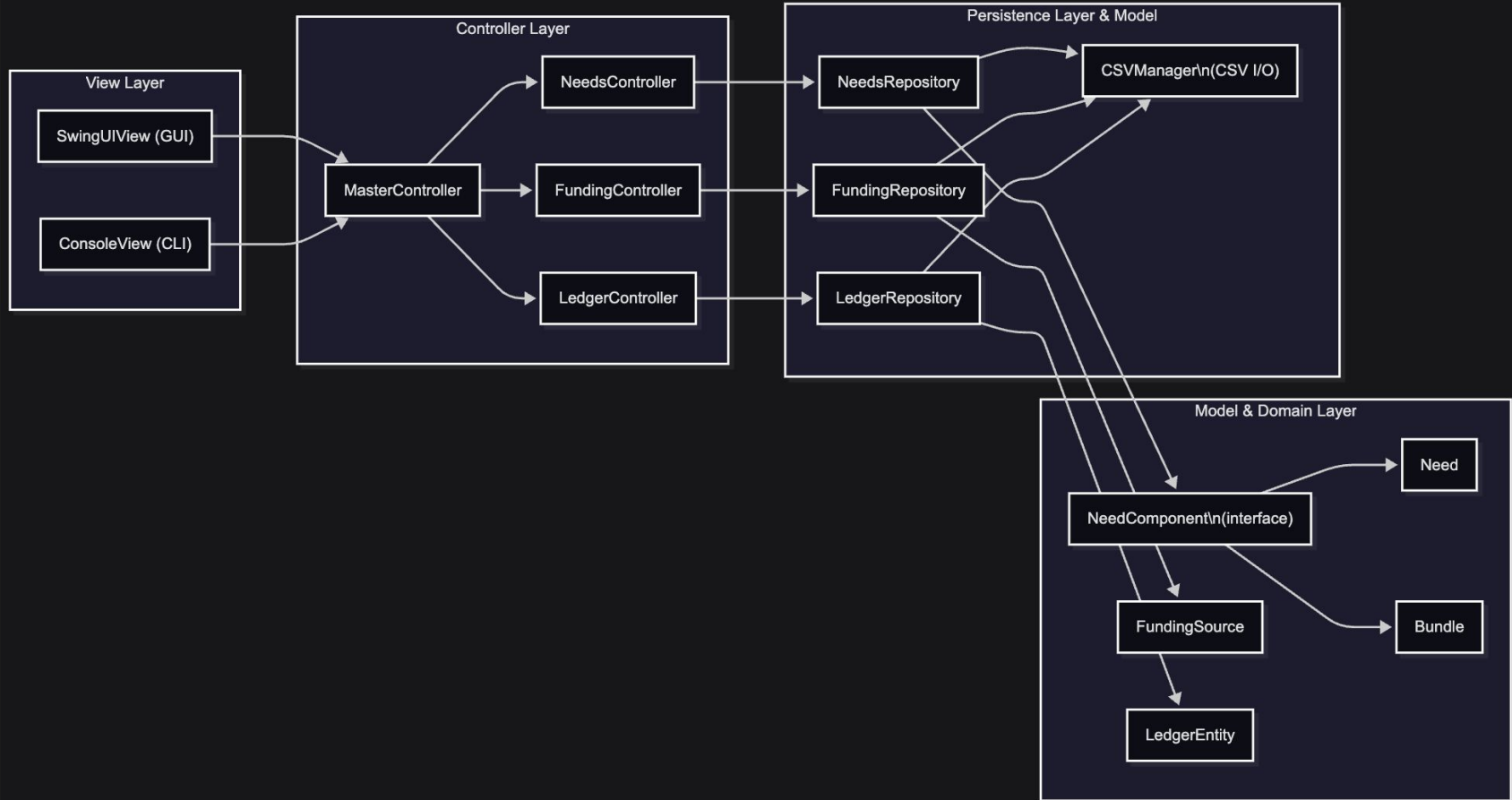


Type	Name	Total Cost (\$)	Components
Need	Gym Membership	45.00	
Need	Electric Bill	150.00	
Need	Internet Service	75.00	
Need	Phone Plan	65.00	
Need	Grocery Budget	400.00	
Need	Water Bill	60.00	
Need	Car Insurance	120.00	
Need	Streaming Subscription	20.00	
Need	Medical Insurance	250.00	
Need	Pet Care	80.00	
Need	Monthly Rent	2500.00	
Bundle	Fitness Package	105.00	Gym Membership (1), Water B...
Bundle	Home Utilities	285.00	Electric Bill (1), Water Bill (1), I...
Bundle	Communication Plan	140.00	Phone Plan (1), Internet Servic...
Bundle	Living Essentials	3110.00	Monthly Rent (1), Grocery Bud...
Bundle	Complete Lifestyle Bundle	4975.00	Living Essentials (1), Grocery B...

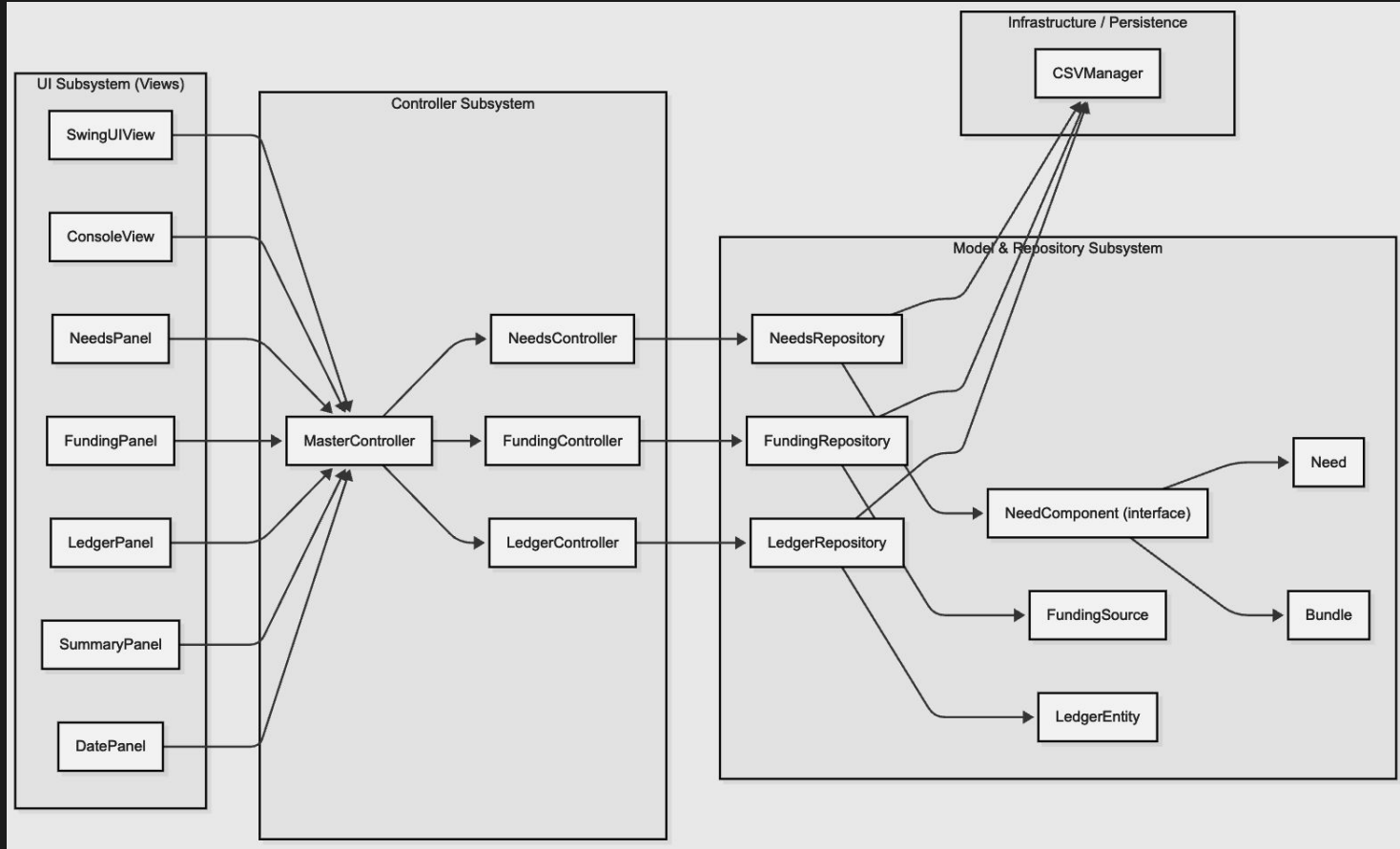
Catalog Actions (List Needs)

Add Need Add Bundle Edit Need Total Edit Bundle Components Delete Need/Bundle

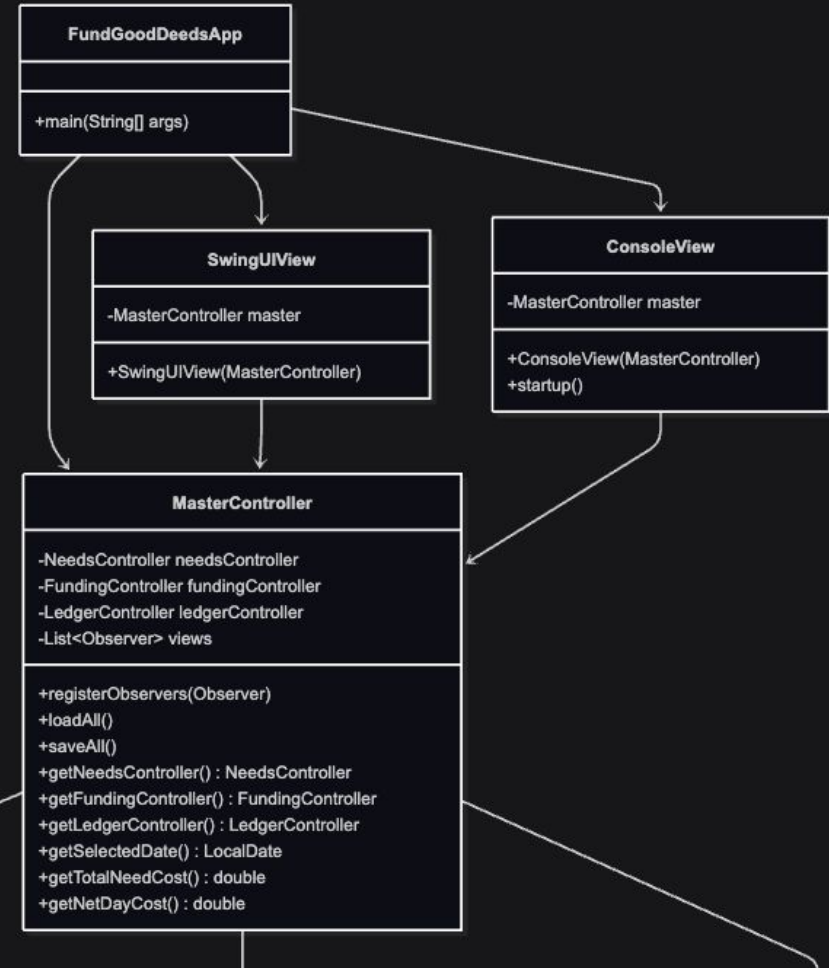
Architecture Overview (MVC + REPOS)



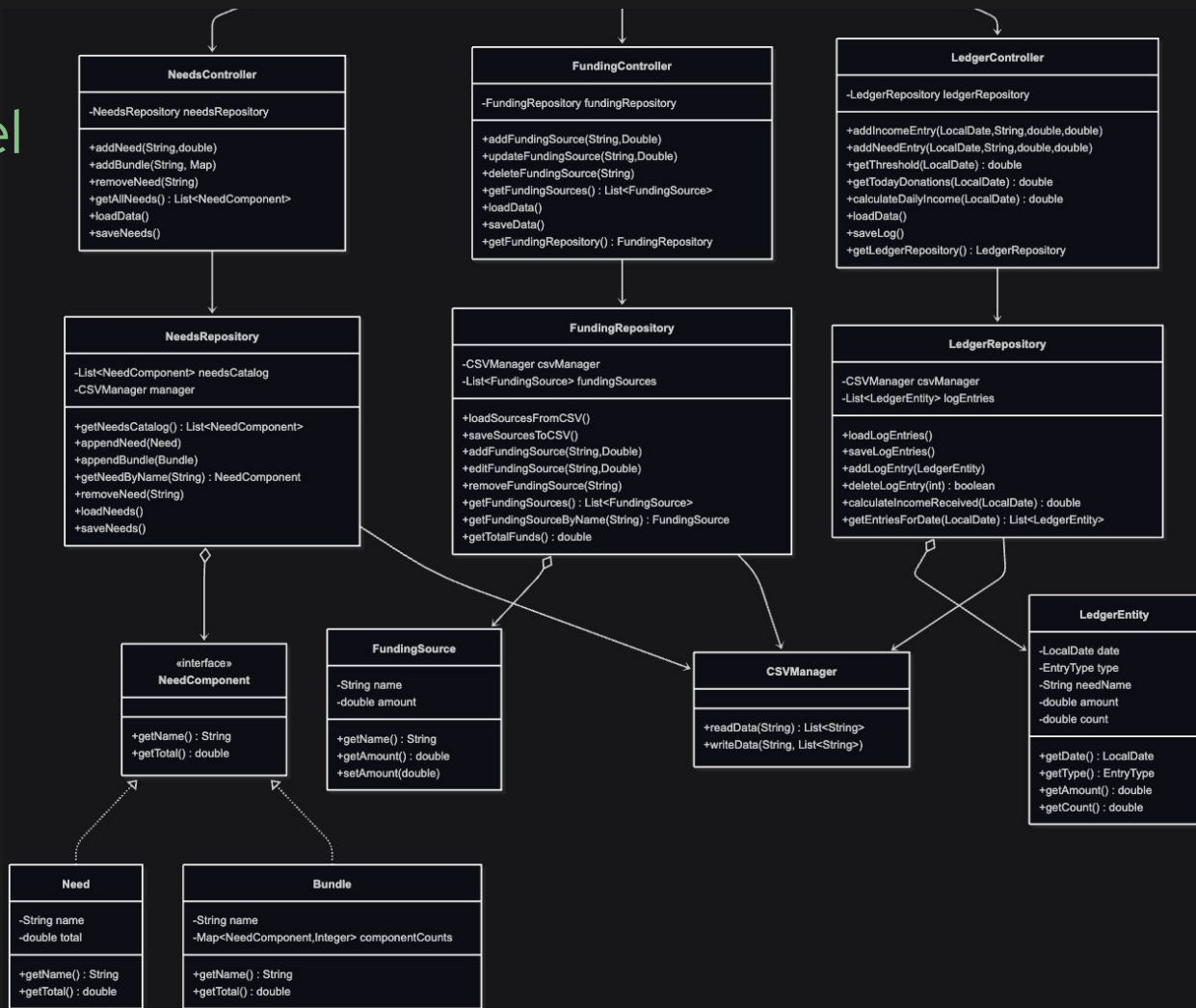
Subsystem Breakdown



View + Controller Relationship

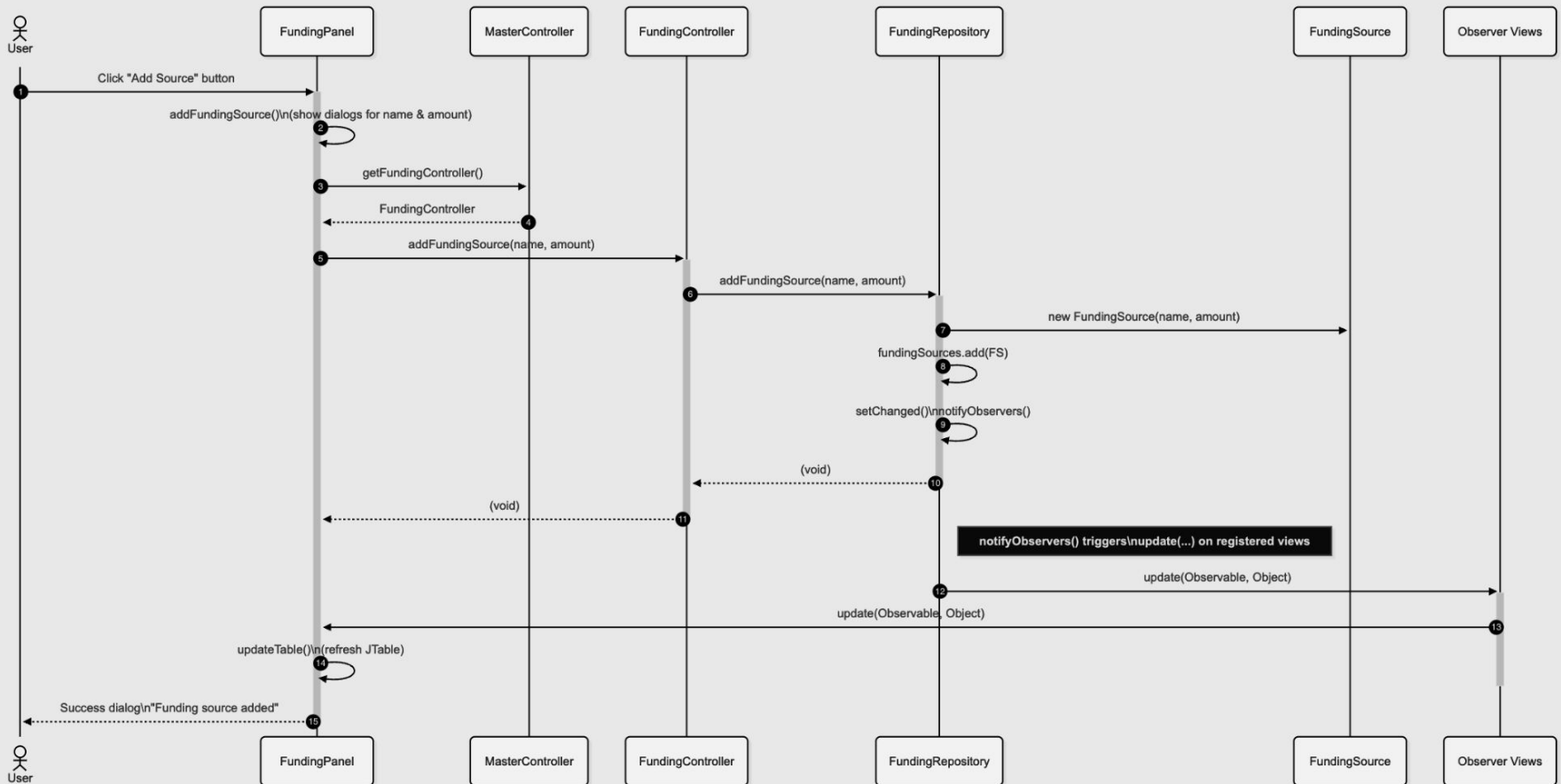


Controller + Model Relationship



Sequence Diagram Add Funding Source

*Observer Views = GUI or CLI classes



Live Demo

Load needs + funding from csv

Add new funding source

Allocate funding

Show updated gui response

Show ui persistence validation

- Open and launch app with maven commands
- Show needs and bundle page
 - Create a need/bundle
 - Could be as simple as (NEEDTEST / BUNDLETEST)
 - Edit a need (name or total cost)/bundle (quantity of a component)
 - Delete a need/bundle
- Move to funding page
 - Add a funding source (FUNDINGTEST or something)
- Move to ledger
 - Add a ledger entry (a need/bundle/funding source) to the point the threshold is exceeded
 - Change to a future date, and then set a daily threshold or something
- Show the summary that the threshold is exceeded
- Can show persistence with exiting and reentering the app upon saving to show it sticks

Challenges We Hit

- Ensuring CSV files matched the Specs
- GUI State Refresh
- Conflicts during integrations (model and controller expectations)
- Learning to work with Master Controller

Principles and patterns We Applied

Patterns

- Observer
 - Updating based on changes in model
- Repository Pattern
 - Storage modularity

Principles

- SRP
- Low Coupling / High Cohesion
 - Clear subsystem lines
- DRY
 - Shared Helper Methods

Improvements Going Forward to R3

- Refactor gui to have less “repetitive” looking code
- Improve UX
- Try as a web service with HTML display
- Users
 - Splitting Bills with people
 - Savings planning for kids educations
- Introduce repository interfaces for DIP

Lessons Learned

- Subsystem Boundaries Matter
- Good Diagrams Prevent Design Drift
- Communication Saves Hours
- MVC only works if used properly
- Integration days always take longer than expected
- Captioning Services are amazing and helped us so much

Q & A