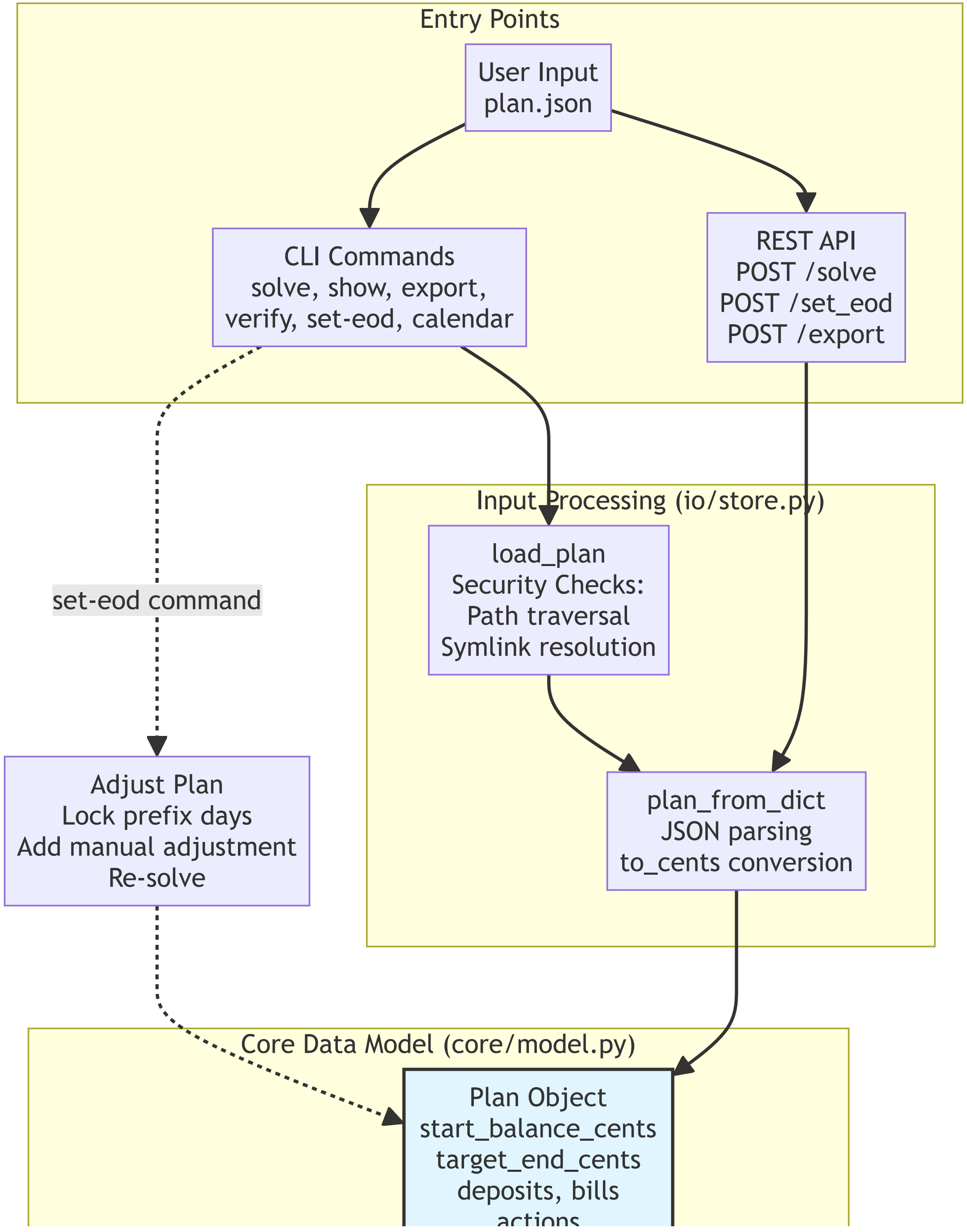
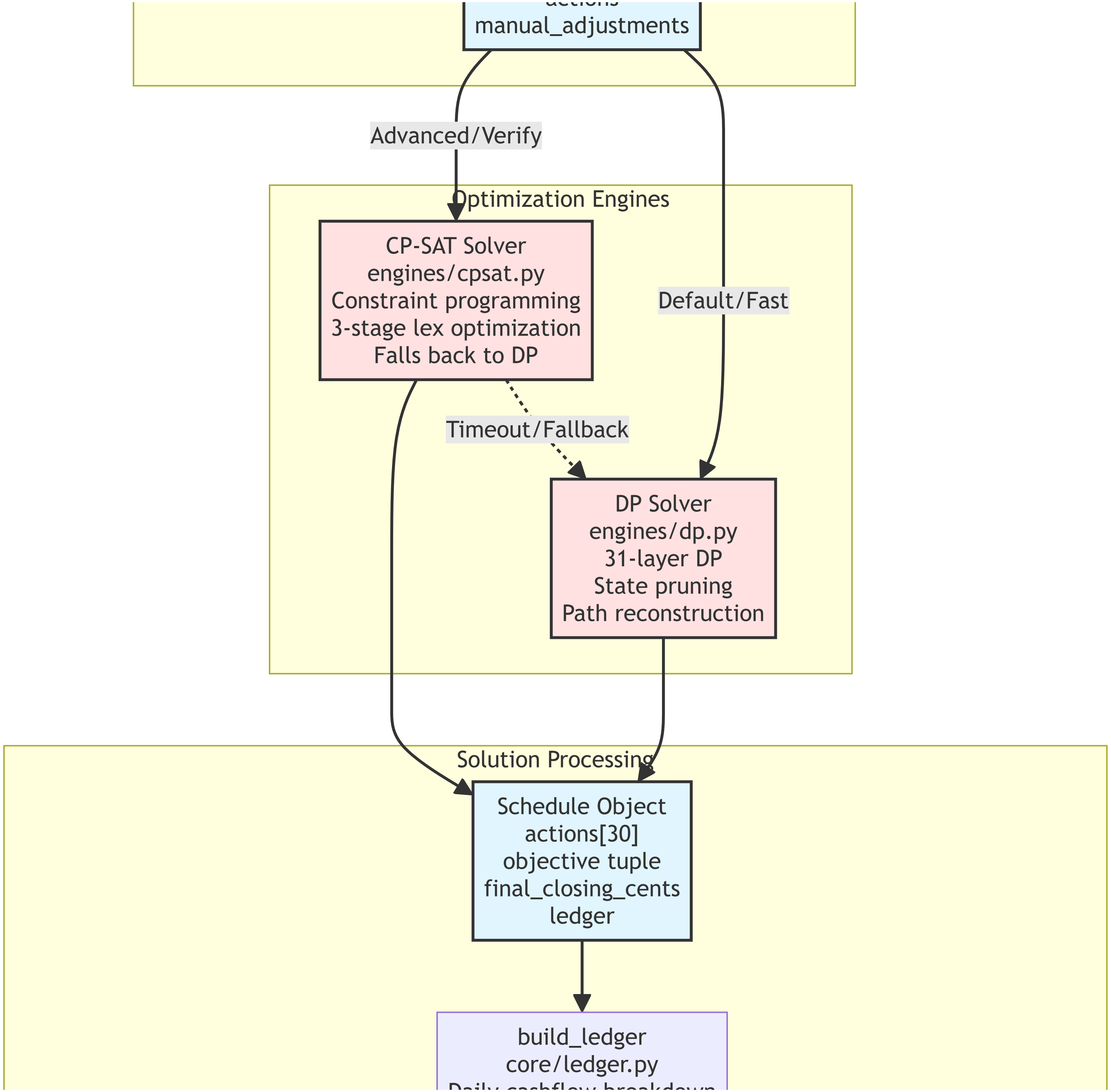
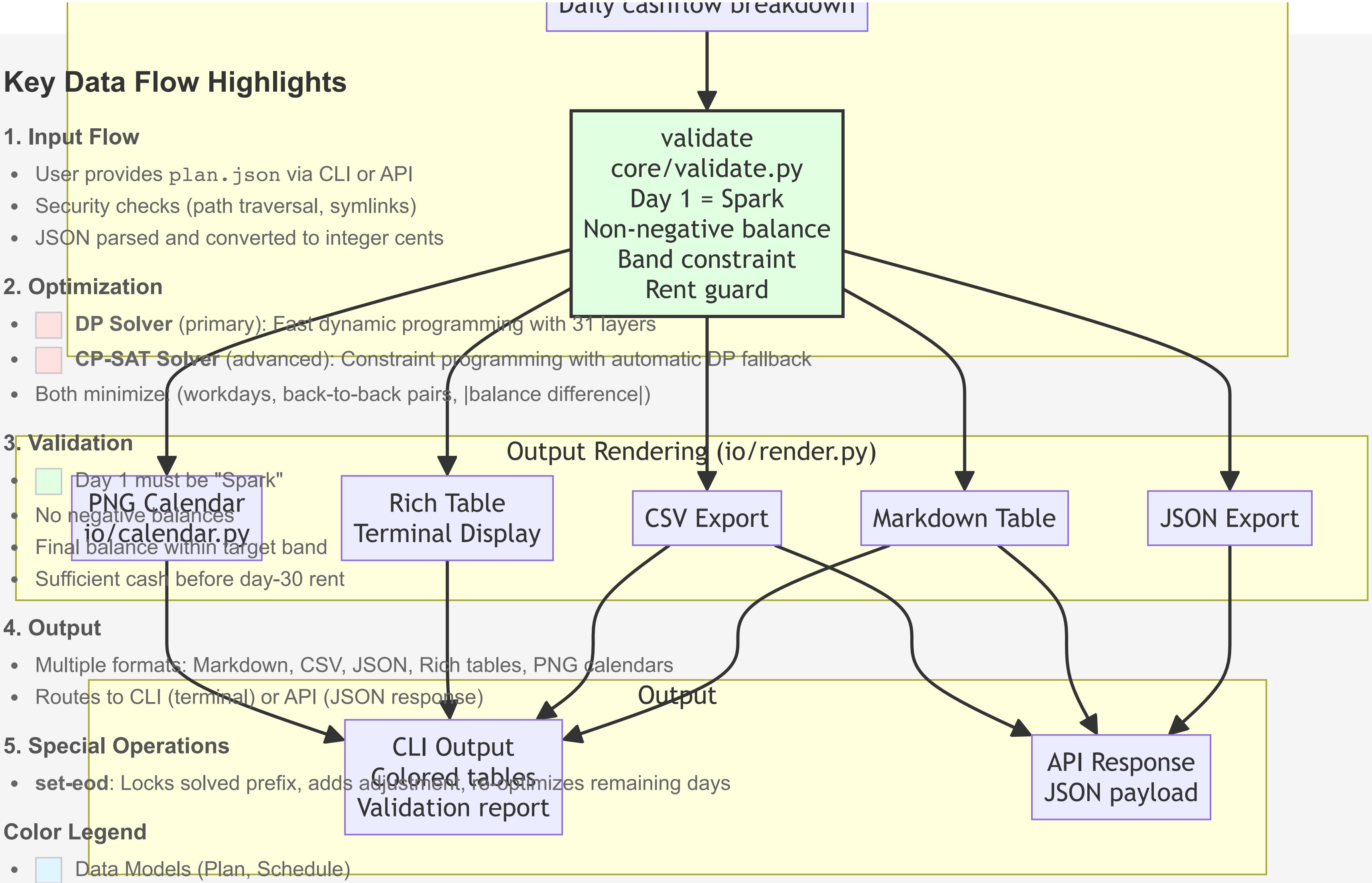
Cashflow Scheduler

Architecture & Data Flow Diagram


Entry Points
User Input
plan.json
CLI Commands
solve, show, export,
verify, set-eod, calendar
REST API
POST /solve
POST /set_eod
POST /export
Input Processing (io/store.py)
load_plan
Security Checks:
Path traversal
Symlink resolution
plan_from_dict
JSON parsing
to_cents conversion
set-eod command
Adjust Plan
Lock prefix days
Add manual adjustment
Re-solve
Core Data Model (core/model.py)
Plan Object
start_balance_cents
target_end_cents
deposits, bills
actions


actions
manual_adjustments
Advanced/Verify
Default/Fast
Optimization Engines
CP-SAT Solver
engines/cpsat.py
Constraint programming
3-stage lex optimization
Falls back to DP
Timeout/Fallback
DP Solver
engines/dp.py
31-layer DP
State pruning
Path reconstruction
Solution Processing
Schedule Object
actions[30]
objective tuple
final_closing_cents
ledger
build_ledger
core/ledger.py
Daily cashflow breakdown



Key Data Flow Highlights

1. Input Flow

- User provides `plan.json` via CLI or API
- Security checks (path traversal, symlinks)
- JSON parsed and converted to integer cents

2. Optimization

- 🟥 **DP Solver** (primary): Fast dynamic programming with 31 layers
- 🟥 **CP-SAT Solver** (advanced): Constraint programming with automatic DP fallback
- Both minimize: (workdays, back-to-back pairs, |balance difference|)

3. Validation

- 🟩 Day 1 must be "Spark"
- No negative balances
- Final balance within target band
- Sufficient cash before day-30 rent

4. Output

- Multiple formats: Markdown, CSV, JSON, Rich tables, PNG calendars
- Routes to CLI (terminal) or API (JSON response)

5. Special Operations

- **set-eod**: Locks solved prefix, adds adjustment, re-optimizes remaining days

Color Legend

- 🟦 Data Models (Plan, Schedule)
- 🟥 Solver Engines (DP, CP-SAT)
- 🟩 Validation