Software Design Documentation

Overview

This risk-calculation system computes 5-day Value at Risk (VaR) and Expected Shortfall (ES) for both stock-only and mixed stock+option portfolios. The entry point is a **main.py** wrapper that lets the user choose between:

1. **Historical calibration** (uses CSV price history to estimate μ/σ)
2. **Manual parameters** (user supplies μ and σ directly)

Each workflow then:

- Prompts for portfolio positions
- Computes parametric and Monte Carlo VaR/ES
- Prints numeric summaries and saves comparison graphs

The core model engines remain in their own modules (`parametric5yr.py`, `parametric_ewm.py`, `historical.py`, `montecarlo.py`, plus the manual-input module).

Input Specification

1. **Mode selection**
    - Historical calibration (from CSV)
    - Manual μ/σ input
2. **Common inputs**
    - **VaR confidence level** (e.g. `0.99`)
    - **ES confidence level** (e.g. `0.975`)
    - **Monte Carlo simulations** (e.g. `10000`)
3. **Historical calibration mode**
    - **CSV file**: date-indexed prices for each stock
    - **Stock positions**: code + number of shares
4. **Manual-input mode**
    - **Stock positions**: code, shares, current price, μ, σ
    - **Option positions**: underlying code, contracts, current price, strike, maturity, μ, σ, (optional) r, q, type

All inputs are collected interactively via prompts.

High-Level Architecture

```
             ┌─────────────────────────────┐
             │          main.py            │
             │  (wrapper: choose mode)     │
             └─────────────┬───────────────┘
                           │
           ┌───────────────┴──────────────┐
           │ historical_calibration.py    │
           │  (CSV → est. μ/σ → run)      │
           └───────────────┬──────────────┘
                           │
           ┌───────────────┴──────────────┐
           │   input_mu_sigma.py          │
           │(manual μ/σ → run param.)     │
           └───────────────┬──────────────┘
                           │
      ┌────────────────────┴──────────────────────┐
      │            model modules                  │
      │ ┌───────────────┐ ┌────────────────────┐  │
      │ │parametric5yr  │ │ parametric_ewm     │  │
      │ └───────────────┘ └────────────────────┘  │
      │ ┌───────────────┐ ┌────────────────────┐  │
      │ │ historical    │ │ montecarlo         │  │
      │ └───────────────┘ └────────────────────┘  │
      │         ┌─────────────────────────────┐   │
      │         │ portfolio_var_es.py         │   │
      │         └─────────────────────────────┘   │
      └──────────────┬──────────────┬─────────────┘
                     │              │
           ┌─────────▼────────┐ ┌───▼───────────┐
           │  Pandas (I/O)    │ │ SciPy/stats   │
           └──────────────────┘ └───────────────┘
                    │                 │
                ┌───▼────────────┐ ┌──▼─────────────┐
                │ NumPy (Sims)   │ │ Matplotlib     │
                └────────────────┘ └────────────────┘
```

Module Interfaces

main.py

```python
def main() -> None:
    """
    1) Prompt user: choose historical vs. manual mode
    2) Delegate to:
       - historical_calibration.main()
       - input_mu_sigma.main()
    """
```

historical_calibration.py

```
def main() -> None:
    """
    1) Prompt for CSV file, VaR/ES levels
    2) Prompt for stock codes & positions
    3) Load price history, build portfolio series
    4) Call each model's compute_var/compute_es
    5) Print summary; save var_comparison.png, es_comparison.png
    """
```

input_mu_sigma.py

```
def main() -> None:
    """
    1) Prompt for stock and option positions + current prices
    2) Prompt for user-provided mu, sigma, VaR/ES levels, MC sims
    3) Compute per-instrument VaR/ES via:
       - parametric_var / parametric_es
       - mc_var / mc_es
       - option_parametric_var / option_parametric_es
       - option_mc_var / option_mc_es
    4) Print numeric results; save manual_var.png, manual_es.png
    """
```

parametric5yr.py / parametric_ewm.py / historical.py / montecarlo.py

Each exposes:

```
def compute_var(prices: pd.Series, var_level: float, ...) -> pd.Series
def compute_es(prices: pd.Series, es_level: float, ...) -> pd.Series
```

(Parameters vary: window size, λ, n_sims.)

Data Flow & Structures

- **Prices DataFrame**
  - Index: `DatetimeIndex` of trading dates
  - Columns: stock price series (CSV mode)
- **Portfolio Series**
  - Weighted sum of individual stock prices (historical mode)
  - Stand-alone series for manual mode uses provided prices

- **Intermediate**
  - Daily and 5-day log-returns (`np.log(P_t/P_{t-1})`)
  - 5-day P&L series
- **Outputs**
  - `pd.Series` of VaR/ES for each method
  - Plots saved as PNGs in `output/`

Graphical Output

- **VaR Comparison** (`var_comparison.png`) – Overlaid time series of all methods' VaR
- **ES Comparison** (`es_comparison.png`) – Overlaid time series of all methods' ES

Generated with `matplotlib`, saved under `output/`.