HCP Tracker Product Requirements Document

Version: 3.4.2
File: hcp_tracker_prd_v3_4_2.md
Last Updated: 2025-08-31 23:55:00 UTC
Status: Active Development with Hybrid Architecture
Current Code Version: v6.4.1 (Steps 1-3 Complete)

---

1. Executive Summary

1.1 Product Vision

The HCP Tracker is a browser-based portfolio optimization tool implementing the Humble Conviction Portfolio (HCP) Investment Policy Statement v3.10. It provides a guided 10-step workflow from investment philosophy understanding through portfolio rebalancing, using a hybrid development architecture that separates development concerns from deployment requirements.

1.2 Key Changes in v3.4.2

- **Hybrid Development Architecture**: Modular development with integrated deployment
- **Complete Steps 1-3**: Philosophy, Data Import/Edit, Theme Analysis fully functional
- **Four Embedded Modules**: FileHandler v1.4, ThemeCalculator v2.9, DataEditor v1.0, Indicators v1.0
- **Production Ready**: Single-file deployment at 108KB with offline capability
- **Enhanced Sample Data**: 5 market scenarios with proper indicator differentiation

1.3 Implementation Status

- **Current Version:** v6.4.1 (2025-08-31 23:50:00 UTC)
- **Functionality:** Steps 1-3 complete, Steps 4-10 in development
- **Data Collector Compatibility:** v3.8+ (with nested theme structure)
- **IPS Version:** v3.10 compliant with Appendix H specifications
- **Browser Support:** Chrome, Firefox, Safari, Edge (ES6+)

---

2. Architecture Overview

2.1 Technical Stack

- **Frontend:** Vanilla JavaScript (ES6+)
- **Styling:** Inline CSS (no external dependencies)
- **Storage:** LocalStorage for state persistence
- **Data Format:** JSON for import/export
- **Deployment:** Single HTML file (~108KB base + embedded modules)

2.2 Core Architecture

```javascript
// Global namespace structure (v6.4.1)
const tracker = {
  version: '6.4.1',
  currentStep: 1,
  completedSteps: [],

  state: {
    // Data sources
    philosophyAcknowledged: false,
    initializationData: null,
    monthlyData: null,
    dataQuality: {},
    manualOverrides: {},

    // Calculated values
    themeProbabilities: {},
    scenarioProbabilities: []
  },

  // Core methods
  init(),
  navigateToStep(n),
  validateStep(n),
  saveState(),
  loadState()
}
```

2.3 Embedded Modules Architecture

```javascript
// Embedded modules available globally
const FileHandler = { version: '1.4', /* methods */ };
const ThemeCalculator = { version: '2.9', /* methods */ };
const DataEditor = { version: '1.0', /* methods */ };
const Indicators = { version: '1.0', /* methods */ };
```

---

3. 10-Step Workflow Implementation

3.1 Workflow Overview

Each step has validation gates preventing forward progress until requirements are met.

3.2 Step Definitions

Step 1: Investment Philosophy ✅ COMPLETE

- **Purpose:** Acknowledge HCP investment framework
- **Requirements:** User must check acknowledgment box
- **Validation:** `state.philosophyAcknowledged = true`
- **Status:** Fully implemented

Step 2: Data Import & Edit ✅ COMPLETE

- **Purpose:** Import data and allow manual overrides
- **Requirements:** Monthly data file or sample generation
- **Features:**
  - File upload for initialization and monthly data
  - 5 sample data scenarios (Tech Boom, USD Strength, P/E Reversion, International, Mixed)
  - Complete data editing with modal system
  - Manual override tracking with yellow highlighting
- **Status:** Fully implemented with embedded modules

Step 3: Theme Analysis ✅ COMPLETE

- **Purpose:** Calculate theme probabilities using IPS v3.10 methodology

- **Features:**
  - Real indicator-based calculations (not random)
  - 13 indicators across 4 themes with three-tier weighting
  - 16-scenario probability matrix generation
  - Enhanced momentum calculations
- **Status:** Fully implemented with embedded ThemeCalculator v2.9

Step 4: Scenario Analysis 🔄 IN DEVELOPMENT

- **Purpose:** Detailed analysis of 16 possible macro scenarios
- **Requirements:** Theme probabilities from Step 3
- **Planned Features:**
  - Scenario ranking and probability distribution
  - Risk-return profiles per scenario
  - Correlation analysis between scenarios

Step 5: Portfolio Optimization 🔄 IN DEVELOPMENT

- **Purpose:** Mean-variance optimization across scenarios
- **Requirements:** Scenario analysis from Step 4
- **Planned Features:**
  - Probability-weighted optimization
  - Risk constraints and bounds
  - Asset allocation recommendations

Step 6: Current Positions 🔄 IN DEVELOPMENT

- **Purpose:** Input current portfolio holdings
- **Requirements:** User manual input
- **Planned Features:**
  - Portfolio position entry interface
  - Current allocation analysis
  - Drift calculation from optimal

Step 7: Rebalancing Trades 🔄 IN DEVELOPMENT

- **Purpose:** Generate specific trades to reach optimal allocation

- **Requirements:** Current positions and optimal allocation
- **Planned Features:**
  - Trade list generation
  - Tax optimization considerations
  - Execution priority ranking

Step 8: History 🔄 IN DEVELOPMENT

- **Purpose:** Historical tracking and audit trail
- **Requirements:** Previous tracker usage
- **Planned Features:**
  - Change log and decision history
  - Performance attribution
  - Scenario accuracy tracking

Step 9: Report 🔄 IN DEVELOPMENT

- **Purpose:** Generate comprehensive analysis report
- **Requirements:** Completed analysis
- **Planned Features:**
  - PDF report generation
  - Executive summary
  - Detailed methodology appendix

Step 10: Export 🔄 IN DEVELOPMENT

- **Purpose:** Export data and results
- **Requirements:** Completed tracker workflow
- **Planned Features:**
  - CSV exports for trades, indicators, scenarios
  - JSON backup of complete state
  - Integration with external portfolio systems

---

4. Critical Display Specifications

4.1 Scenario Probability Color Coding

The 16-scenario matrix uses a 5-tier color system based on probability ranges:

Probability Range	Color	CSS Class	Hex Code	Description
> 25%	Dark Green	scenario-very-high	#155724	Extremely likely scenarios
10-25%	Light Green	scenario-high	#28a745	Likely scenarios
5-10%	Yellow	scenario-medium	#ffc107	Moderate probability
1-5%	Light Red	scenario-low	#dc3545	Unlikely scenarios
< 1%	Dark Red/Gray	scenario-very-low	#6c757d	Extremely unlikely

4.2 Theme Color Assignments

Fixed theme colors for consistency across all displays:

Theme	Color Name	CSS Class	Hex Code	RGB
USD	Red	theme-usd	#dc3545	rgb(220, 53, 69)
AI/Innovation	Blue	theme-ai	#007bff	rgb(0, 123, 255)
P/E	Yellow	theme-pe	#ffc107	rgb(255, 193, 7)
International	Green	theme-intl	#28a745	rgb(40, 167, 69)

4.3 Data Confidence Indicators

Separate from probability colors, data confidence shows quality of underlying data:

Confidence Level	When to Use	Display
HIGH	All indicators fresh, complete history	Green dot
MEDIUM	Some stale data or interpolation	Yellow dot
LOW	Significant missing data	Red dot

---

5. Three-Tier Signal Framework

5.1 Fixed Tier Weights

- **Canary**: 35% (early warning)
- **Primary**: 40% (core signals)
- **Structural**: 25% (confirmation)

5.2 Indicator Classification

Tier	Indicators	Weight	Per-Indicator Weight
Canary	DXY, QQQ/SPY, Risk Premium, ACWX/SPY	35%	8.75% each
Primary	Forward P/E, Net Margins, Yuan SWIFT, CAPE	40%	10% each
Structural	Productivity, Reserve Share, Gold Purchases, TIC Flows	25%	6.25% each

---

6. UI Display Requirements

6.1 Step 3 Theme Display

Each theme shows:

- Theme name with theme color bar
- Percentage probability (large, bold)
- NO confidence labels on probabilities
- Theme color fill proportional to probability

6.2 Scenario Matrix Display

- All 16 scenarios in binary order (0000-1111)
- Rank numbers and colors change based on probabilities
- Binary code for each scenario
- Probability percentage with appropriate color coding

---

7. Data Requirements

7.1 Input Data Structure (FileHandler v1.4 Compatible)

```javascript
```

```
{
  "data_type": "monthly",
  "version": "3.8.2",
  "timestamp": "2025-08-31T23:50:00.000Z",
  "indicators": {
    "usd": {
      "dxy": { "current": 103.45, "history": [450 data points] },
      "gold_purchases": { "current": 35.8, "history": [...] },
      "yuan_swift": { "current": 4.74, "history": [...] },
      "reserve_share": { "current": 58.4, "history": [...] }
    },
    "innovation": {
      "qqq_spy": { "current": 0.756, "history": [...] },
      "productivity": { "current": 2.3, "history": [...] },
      "net_margins": { "current": 12.0, "history": [...] }
    },
    "pe": {
      "forward_pe": { "current": 20.8, "history": [...] },
      "cape": { "current": 34.2, "history": [...] },
      "risk_premium": { "current": 0.62, "history": [...] }
    },
    "intl": {
      "acwx_spy": { "current": 0.96, "history": [...] },
      "sp_vs_world": { "current": 1.034, "history": [...] },
      "tic_flows": { "current": 125.4, "history": [...] }
    }
  }
}
```


7.2 Sample Data Scenarios

FileHandler v1.4 generates 5 distinct scenarios:

- **Tech Boom**: Strong AI indicators (QQQ/SPY 0.82, Productivity 3.8%)
- **USD Strength**: High DXY (108.5), strong reserve share (61.2%)
- **P/E Reversion**: Elevated valuations (Forward P/E 24.5, CAPE 38.8)
- **International**: Weak USD enabling international outperformance
- **Mixed/Random**: Balanced or randomized signals across themes

---

8. User Interactions

8.1 Primary Actions

- **Step Navigation**: Click progress indicators or Previous/Next buttons
- **Sample Data Generation**: Click scenario buttons (Tech Boom, USD Strength, etc.)
- **Data Editing**: Click Edit buttons to open modal for manual overrides
- **Philosophy Acknowledgment**: Check box to proceed to Step 2

8.2 Data Editing Flow

1. Click Edit button → Modal appears with current value
2. Enter new value and select reason for override
3. Save → Row highlights in yellow, theme analysis recalculates
4. Manual overrides persist in state and localStorage

8.3 Navigation Rules

- **Step 1** → **2**: Requires philosophy acknowledgment
- **Step 2** → **3**: Requires monthly data (file or sample)
- **Step 3** → **4**: Requires theme analysis completion
- **Steps 4-10**: Currently locked, will require prior step completion

---

9. Data Persistence

9.1 LocalStorage Keys

- `hcp-tracker-v641-state`: Main application state
- Includes version, current step, completed steps, and full state object

9.2 State Structure

```javascript
```

```
{
  version: '6.4.1',
  currentStep: 1,
  completedSteps: [],
  state: {
    philosophyAcknowledged: false,
    initializationData: null,
    monthlyData: null,
    manualOverrides: {},
    themeProbabilities: {},
    scenarioProbabilities: []
  }
}
```

---

10. Validation and Error Handling

10.1 File Upload Validation

- JSON parsing with error messages
- File naming convention checking
- Data structure validation

10.2 Step Validation

- Philosophy: Checkbox must be checked
- Data: Monthly data must be loaded
- Analysis: Theme calculations must complete successfully

10.3 Manual Override Validation

- Numeric values required
- Reason selection mandatory
- Notes optional but tracked

---

11. Performance and Technical Specifications

11.1 File Size Targets

- **Current (v6.4.1)**: ~108KB total

- **Projected Final**: ~200KB with all modules
- **Optimization Options**: Minification (30-40% reduction possible)

11.2 Browser Compatibility

- **ES6+ Required**: Arrow functions, const/let, template literals
- **LocalStorage**: Required for state persistence
- **JSON Support**: Native JSON parsing/stringify
- **No External Dependencies**: Fully self-contained

11.3 Performance Characteristics

- **Initialization**: < 100ms on modern browsers
- **Theme Calculation**: < 50ms with 450 data points
- **Modal Operations**: Immediate response
- **Step Navigation**: Instantaneous with state caching

---

12. Hybrid Development Architecture (NEW)

12.1 Development Methodology

The HCP Tracker uses a **hybrid development approach** that separates development concerns from deployment requirements:

Development Phase:

- Individual modules as separate `.js` files
- Clean separation of concerns
- Independent version control per module
- Easy testing and debugging

Deployment Phase:

- Single HTML file with embedded modules
- No external dependencies
- Offline functionality
- Single-file distribution

12.2 Module Architecture

12.2.1 Core Modules (Developed)

- **FileHandler v1.4** (`file_handler_v1_4.js`)
  - Sample data generation with 5 market scenarios
  - 13 indicators with 450 data points for MA calculations
  - Nested theme structure: `{usd: {dxy: {...}}, innovation: {...}}`
- **ThemeCalculator v2.9** (`theme_calculator_v2_9.js`)
  - IPS v3.10 Appendix H Mathematical Specifications
  - Enhanced transition probability calculations
  - 16-scenario probability matrix generation
- **DataEditor v1.0** (`data_editor_v1_0.js`)
  - Modal-based indicator editing
  - Manual override tracking with yellow highlighting
  - Validation and reason logging
- **Indicators v1.0** (`indicators_v1_0.js`)
  - 13 indicator definitions across 4 themes
  - Three-tier framework (canary, primary, structural)
  - Proper weighting and tier calculations

12.2.2 Planned Modules

- **Optimizer v1.0** - Mean-variance optimization for 16 scenarios
- **PositionManager v1.0** - Current portfolio input and validation
- **TradeGenerator v1.0** - Rebalancing trade calculation
- **ReportGenerator v1.0** - PDF/CSV export functionality

12.3 Integration Strategy

12.3.1 Development Workflow

1. **Modular Development:**

```
/modules/
  ├── file_handler_v1_4.js
  ├── theme_calculator_v2_9.js
  ├── data_editor_v1_0.js
  └── indicators_v1_0.js
```

2. **Integration Testing:**

```
/integration_tests/
  └── integration_test_v3_2_1.html  # Loads via <script> tags
```

3. **Deployment Build:**

```
/deployment/
  └── hcp_tracker_v6_4_1_integrated.html  # Embedded modules
```

12.3.2 Embedding Process

Modules are embedded in deployment builds using:

```javascript
// Module embedded as const object
const FileHandler = {
    version: '1.4',
    // ... full module code
};
```

12.4 Version Control Strategy

12.4.1 Module Versioning

Each module maintains independent semantic versioning:

- **MAJOR**: Breaking API changes
- **MINOR**: New features, backward compatible
- **PATCH**: Bug fixes

12.4.2 Integration Versioning

Deployment builds use composite versioning:

- Base system version (e.g., v6.4.1)

- Module compatibility matrix
- Integration test validation

12.4.3 Development History Tracking

```yaml
v6.4.1_integrated:
  base: v6.4.0_modular
  modules:
    FileHandler: v1.4
    ThemeCalculator: v2.9
    DataEditor: v1.0
    Indicators: v1.0
  functionality: "Steps 1-3 complete"
  status: "Production ready"
```

12.5 Benefits of Hybrid Approach

12.5.1 Development Benefits

- **Modularity**: Clean separation of concerns
- **Testability**: Individual module testing
- **Maintainability**: Isolated bug fixes and features
- **Collaboration**: Multiple developers can work on different modules
- **Code Reuse**: Modules can be shared across projects

12.5.2 Deployment Benefits

- **Single File**: No dependency management
- **Offline Support**: Works without internet connection
- **Distribution**: Easy sharing and hosting
- **Performance**: No HTTP requests for module loading
- **Security**: No external script injection points

12.6 Quality Assurance

12.6.1 Module Testing

- Unit tests for individual module functions
- Integration tests for module compatibility

- Scenario-based validation (Tech Boom, USD Strength, etc.)

12.6.2 Integration Validation

- All modules load without conflicts
- Proper data flow between modules
- UI/UX consistency across integrated system
- Performance benchmarking

---

13. Development Guidelines

13.1 Code Standards

- **Naming:** camelCase for functions, UPPER_CASE for constants
- **Comments:** JSDoc format for functions
- **Versioning:** Semantic versioning (major.minor.patch)
- **Module Independence:** Each module must be self-contained

13.2 Release Process

1. Develop module as standalone `.js` file
2. Create integration tests with existing modules
3. Embed in deployment HTML file
4. Test end-to-end functionality
5. Update version numbers and documentation
6. Update PRD if architectural changes made

13.3 File Naming Convention

```
Development Modules: {module_name}_v{major}_{minor}.js
Integration Tests: integration_test_v{major}_{minor}_{patch}.html
Deployment Builds: hcp_tracker_v{major}_{minor}_{patch}_integrated.html
PRD Updates: hcp_tracker_prd_v{major}_{minor}_{patch}.md
```

---

14. Future Development

14.1 Immediate Priorities (Steps 4-6)

- **Optimizer v1.0**: Mean-variance optimization module
- **PositionManager v1.0**: Portfolio input interface
- **TradeGenerator v1.0**: Rebalancing calculation engine

14.2 Version 7.0 Vision

- Complete 10-step workflow
- PDF report generation
- Advanced optimization constraints
- Real-time data integration capabilities

14.3 Long-term Enhancements

- **Cloud Integration**: Optional cloud sync and sharing
- **Multi-Portfolio Support**: Manage multiple strategies
- **Performance Attribution**: Track strategy effectiveness
- **Mobile Optimization**: Responsive design improvements

---

15. Implementation Status Summary

15.1 Current Capabilities (v6.4.1)

✅ **Fully Functional Steps 1-3:**

- Complete investment philosophy workflow
- Real sample data generation (5 scenarios)
- Professional data editing with manual overrides
- Actual theme analysis using IPS v3.10 calculations
- 16-scenario probability matrix generation

15.2 Technical Achievements

- **Single-file deployment** with embedded modules
- **108KB total size** for production-ready system
- **Offline capability** with full functionality

- **Real calculations** replacing placeholder/random data
- **Professional UI** with modal editing system

15.3 Next Development Cycle

- **Target**: Steps 4-6 completion in v6.5
- **Architecture**: Continue hybrid development approach
- **Timeline**: Iterative development with integrated releases

---

End of PRD v3.4.2 - Complete with Hybrid Development Architecture