HCP Portfolio Tracker Technical Specification

Version: 1.3
File: hcp_tracker_technical_spec_v1.3.md
Last Updated: 2025-09-02 23:15:00 UTC
Status: Production Ready - Rebalancing Implementation Focus
Target Audience: Developers, Technical Implementers

NEW IN v1.3: Surgical updates for Steps 6-7 rebalancing implementation. Position limits marked as "under consideration," tax optimization marked as "out of scope," and trade execution simplified while preserving all specifications for future enhancement.

Version Compatibility Matrix

Component	Current Series	Compatibility Notes
Tracker Release	**6.5.5+ series**	Production stable with Step 4 fixes
Core Architecture	TrackerCore v1.x	Foundation module
Theme Calculator	**v2.10+ series**	IPS v3.11 compliant with display enhancements
File Handler	v1.5+ series	Momentum-aware generation
Data Collector	v3.8+ series	Independent system

---

1. System Architecture

1.1 Core Design Principles

Modular Architecture: Each functional area implemented as independent module with versioned API
Single-File Deployment: All modules embedded in HTML for zero-dependency operation
Browser-Native: Pure JavaScript/HTML/CSS with no external libraries or frameworks
State Persistence: localStorage-based state management with JSON serialization
Progressive Enhancement: Graceful degradation when features unavailable

1.2 Module Dependencies

```
TrackerCore v1.x (Foundation)
├── Navigation and state management
├── Step validation and workflow
└── Browser storage abstraction
```

FileHandler v1.5+ (Data Layer)
├── Sample data generation with momentum patterns
├── File validation and parsing
└── Nested theme data structure support

ThemeCalculator v2.10+ (Analysis Engine)
├── IPS v3.11 mathematical compliance
├── Enhanced trigger detection algorithms
├── 16-scenario probability generation
├── Current scenario identification
└── Corrected probability color mapping

DataEditor v1.0+ (User Interface)
├── Modal-based editing system
├── Manual override tracking
└── Data table display and validation

PortfolioOptimizer v2.0+ (Optimization Engine)
├── Regret minimization framework
├── Dual optimization methodology
├── Smart hedging protocols
└── 🔄 Position limits under consideration

RebalancingModule v1.0+ (NEW - Trade Generation)
├── Current position input interface
├── Drift calculation and analysis
├── Trade generation with PIMIX/PYLD rules
└── 📋 Simplified execution framework (complex features out of scope)

Indicators v1.0+ (Configuration)
├── 13-indicator framework definitions
├── Three-tier signal classification
└── Theme organization and weighting

---

2. Data Architecture

2.1 State Structure

javascript

```
TrackerCore.state = {
  // Step validation
  philosophyAcknowledged: boolean,

  // Data layer
  monthlyData: {
    indicators: {
      usd: { dxy: {current, history, ...}, ... },
      innovation: { qqq_spy: {current, history, ...}, ... },
      pe: { forward_pe: {current, history, ...}, ... },
      intl: { acwx_spy: {current, history, ...}, ... }
    },
    trading_status: string,
    timestamp: string
  },

  // Analysis results
  themeProbabilities: { usd: number, ai: number, pe: number, intl: number },
  scenarioProbabilities: Array<{id, name, probability, binary, themes}>,
  optimizedAllocation: { [security]: percentage },

  // NEW v1.3 - Rebalancing state
  currentPositions: { [security]: {shares, value, percentage} },
  driftAnalysis: { [security]: {current, target, drift, dollarDrift} },
  rebalancingTrades: Array<{security, action, shares, dollarAmount, reason}>,

  // User modifications
  manualOverrides: { [indicatorKey]: {value, reason, timestamp} },

  // Metadata
  lastDataGeneration: string,
  dataScenario: string,
  calculationResults: object,
  lastRebalancingCalc: string
}
```

2.2 Indicator Data Format

Each indicator follows standardized structure:

```
javascript
```

```
{
  current: number, // Current value
  history: number[], // Historical values (6+ periods minimum)
  freshness: 'fresh'|'stale', // Data quality indicator
  source: string, // Data source identifier
  name: string, // Display name
  manual_override?: boolean, // Manual edit flag
  override_reason?: string,  // Edit justification
  override_timestamp?: string // Edit timestamp
}
```

2.3 NEW v1.3 - Position and Trade Data Formats

Current Position Format:

```
javascript

{
  security: string, // Ticker symbol
  shares: number, // Current share count
  value: number, // Current dollar value
  percentage: number, // Portfolio percentage
  lastUpdated: string // ISO timestamp
}
```

Trade Recommendation Format:

```
javascript

{
  security: string, // Ticker symbol
  action: 'BUY'|'SELL', // Trade direction
  shares: number, // Share quantity
  dollarAmount: number, // Approximate dollar value
  reason: string, // Reason for trade
  priority: 'HIGH'|'MEDIUM'|'LOW', // Execution priority
  constraints: Array<string> // Applied rules (e.g., "PIMIX_HOLD_ONLY")
}
```

---

3. Calculation Algorithms

3.1 Theme Probability Calculation

Algorithm: IPS v3.11 Enhanced Transition Probability Framework

Key Innovation: Dual approach based on indicator state:

- **Triggered indicators:** Signal strength methodology (higher probability range)
- **Non-triggered indicators:** Time-to-trigger methodology (traditional approach)

Mathematical Framework:

```
For triggered indicators:
  base_probability = 0.70  # Higher baseline
  distance_bonus = min(0.30, abs(distance_to_trigger) * 3)
  momentum_boost = favorable_momentum * 0.25
  result = base_probability + distance_bonus + momentum_boost

For non-triggered indicators:
  months_to_trigger = abs(distance_to_trigger) / momentum_rate
  base_probability = time_decay_function(months_to_trigger)
  direction_adjustment = momentum_direction_factor
  result = base_probability * direction_adjustment
```

3.2 Momentum Calculation

Critical Implementation: 6-period baseline comparison

```javascript
calculateMomentum(indicator) {
  const current = indicator.current;
  const baseline = indicator.history[indicator.history.length - 6];
  const percentChange = (current - baseline) / Math.abs(baseline);
  return Math.max(-1, Math.min(1, percentChange * 10));
}
```

3.3 NEW v1.3 - Drift and Trade Calculation Algorithms

Portfolio Drift Calculation:

```javascript
```

```
calculateDrift(currentPositions, targetAllocation, totalPortfolioValue) {
  const driftAnalysis = {};

  for (const security of ALL_SECURITIES) {
    const currentPercent = (currentPositions[security]?.value || 0) / totalPortfolioValue;
    const targetPercent = targetAllocation[security] || 0;
    const driftPercent = currentPercent - targetPercent;
    const dollarDrift = driftPercent * totalPortfolioValue;

    driftAnalysis[security] = {
      current: currentPercent,
      target: targetPercent,
      drift: driftPercent,
      dollarDrift: dollarDrift,
      action: Math.abs(driftPercent) > 0.01 ? (driftPercent > 0 ? 'SELL' : 'BUY') : 'HOLD'
    };
  }

  return driftAnalysis;
}
```

Trade Generation Algorithm:

javascript

```
generateTrades(driftAnalysis, currentPrices, constraints) {
  const trades = [];

  // Apply PIMIX hold-only rule
  if (driftAnalysis.PIMIX.drift > 0) {
    trades.push({
      security: 'PIMIX',
      action: 'SELL',
      dollarAmount: Math.abs(driftAnalysis.PIMIX.dollarDrift),
      reason: 'Reduce income allocation (PIMIX hold-only)',
      priority: 'HIGH',
      constraints: ['PIMIX_HOLD_ONLY']
    });
  }

  // Apply PYLD primary income rule
  const totalIncomeBuys = calculateTotalIncomeBuys(driftAnalysis);
  if (totalIncomeBuys > 0) {
    trades.push({
      security: 'PYLD',
      action: 'BUY',
      dollarAmount: totalIncomeBuys,
      reason: 'Increase income allocation (PYLD primary)',
      priority: 'MEDIUM',
      constraints: ['PYLD_PRIMARY_INCOME']
    });
  }

  // Generate remaining trades ignoring drifts < 1% unless cash would go negative
  // [Additional trade generation logic...]

  return trades;
}
```

3.4 Moving Average Specifications

Frequency-Matched Calculations:

- Daily indicators: Use calendar days for MA periods
- Monthly indicators: Use month-end standardized values
- Quarterly indicators: Interpolation for monthly alignment

Example Specifications:

- DXY: 200D MA vs 400D MA comparison
- QQQ/SPY: 50D MA vs 200D MA comparison
- Forward P/E: 12M MA vs 36M MA comparison
- Productivity: 2Q MA vs 6Q MA comparison

---

4. Module Integration Patterns

4.1 Core Integration Flow

```javascript
// 1. TrackerCore initialization
TrackerCore.init() → {
  loadState(),
  setupEventListeners(),
  navigateToStep(currentStep)
}

// 2. Data loading integration
FileHandler.generateSampleData() →
TrackerCore.processFileHandlerData() → {
  validateFileHandlerData(),
  displayDataEditor(),
  triggerThemeCalculation()
}

// 3. Analysis integration
ThemeCalculator.calculateThemeAnalysis() →
TrackerCore.state.themeProbabilities →
displayResults()

// 4. NEW v1.3 - Rebalancing integration
PortfolioOptimizer.optimizePortfolio() →
TrackerCore.state.optimizedAllocation →
RebalancingModule.calculateDrift() →
RebalancingModule.generateTrades() →
displayTradeRecommendations()
```

4.2 Event-Driven Architecture

State Change Events:

- Philosophy acknowledgment → Enable step 2 navigation
- Data load → Validate and trigger analysis
- Theme calculation → Enable step 4 navigation
- Portfolio optimization → Enable step 6 navigation
- Position input → Calculate drift and enable step 7 navigation
- Manual override → Update calculations and display

Error Handling Pattern:

```javascript
try {
  const result = moduleFunction(data);
  if (result.error) {
    displayError(result.error);
    return false;
  }
  updateState(result);
  return true;
} catch (error) {
  logError(error);
  displayFallbackUI();
  return false;
}
```

---

5. Performance Specifications

5.1 Memory Management

State Size Limits:

- Maximum indicator history: 450 data points per indicator
- State object target: <2MB serialized
- localStorage quota: Monitor and warn at 80% usage

Garbage Collection:

- Clear intermediate calculation objects
- Debounce user input events

- Lazy load non-critical display elements

5.2 Computation Optimization

Theme Calculation Performance:

- Target: <200ms for full 4-theme analysis
- Caching: Store intermediate MA calculations
- Parallelization: Process themes independently where possible

NEW v1.3 - Rebalancing Calculation Performance:

- Target: <100ms for drift calculation across 12 securities
- Target: <150ms for trade generation with all constraints
- Caching: Store current positions and prices for recalculation

5.3 File Size Management

Current Production Metrics:

- Core HTML file: ~150KB (acceptable threshold: <200KB)
- Embedded modules add ~35KB total
- NEW v1.3: Rebalancing module adds ~15KB
- Target deployment size: <250KB total

---

6. Critical Display Specifications

6.1 Scenario Probability Color System - PRESERVED v1.2

5-Tier Color Classification (v6.5.5 Standard):

Probability Range	Color	CSS Class	Hex Code	RGB	Usage
≥ 25%	**Dark Green**	scenario-very-high	#155724	rgb(21, 87, 36)	Extremely likely scenarios
10-25%	**Green**	scenario-high	#28a745	rgb(40, 167, 69)	Likely scenarios
5-10%	**Yellow**	scenario-medium	#ffc107	rgb(255, 193, 7)	Moderate probability
1-5%	**Light Red**	scenario-low	#dc3545	rgb(220, 53, 69)	Unlikely scenarios
< 1%	**Dark Red**	scenario-very-low	#6c757d	rgb(108, 117, 125)	Extremely unlikely

6.2 Theme Color Assignments

Fixed theme colors for consistency across all displays:

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

6.3 NEW v1.3 - Rebalancing Display Specifications

Step 6 - Current Positions Display:

- Position table with security, shares, value, percentage columns
- Color coding: Green (underweight), Red (overweight), Gray (on target)
- Drift indicators: Arrow icons showing direction and magnitude
- Total portfolio value prominently displayed

Step 7 - Trade Recommendations Display:

- Trade table with security, action, shares, dollar amount, reason columns
- Priority coding: High (red background), Medium (yellow), Low (gray)
- Constraint indicators: Icons showing applied rules (PIMIX hold-only, PYLD primary)
- Net cash impact calculation and display
- PIMIX/PYLD rules clearly highlighted with explanatory text

Position Drift Color System:

```css
.drift-positive { color: #dc3545; background-color: #f8d7da; } /* Overweight - red */
.drift-negative { color: #155724; background-color: #d4edda; } /* Underweight - green */
.drift-minimal { color: #6c757d; background-color: #e2e3e5; } /* On target - gray */
.drift-ignored { color: #856404; background-color: #fff3cd; } /* <1% ignored - yellow */
```

6.4 Data Confidence Indicators

Quality indicators separate from probability colors:

Confidence Level	When to Use	Display	Implementation
HIGH	All indicators fresh, complete history	Green dot	`<span style="color: #28a745;">●</span>`
MEDIUM	Some stale data or interpolation	Yellow dot	`<span style="color: #ffc107;">●</span>`
LOW	Significant missing data	Red dot	`<span style="color: #dc3545;">●</span>`

6.5 UI Display Requirements - ENHANCED v1.3

Step 3 Theme Display Standards:

- Theme name with theme color bar
- Percentage probability (large, bold typography)
- **NO confidence labels on probabilities** (per PRD v4.0)
- Theme color fill proportional to probability

Step 4 Scenario Display Standards:

- **All 16 scenarios displayed in BINARY ORDER (S1-S16)** - NOT probability ranking
- **Current scenario identification and highlighting:**
  - Blue border around current scenario card
  - "Current Scenario" label at top of card
  - Enhanced visual prominence (scale, shadow effects)
  - Summary section shows both "Current Scenario" and "Most Likely"
- **Correct probability color coding** per Section 6.1
- **Scenario information display:**
  - Scenario ID (S1-S16) prominently displayed
  - Probability ranking (#1-#16 by likelihood)
  - Binary representation (0000-1111)

- Active themes clearly indicated
- Color-coded background per probability range

NEW Step 6 Position Display Standards:

- Current positions in sortable table format
- Percentage allocations with drift indicators
- Color-coded drift visualization per Section 6.3
- Total portfolio value and cash position clearly displayed
- 🔄 **Position limit warnings under consideration** (not enforced)

NEW Step 7 Trade Display Standards:

- Trade recommendations with execution priority
- PIMIX/PYLD rule enforcement clearly indicated
- Dollar amounts and share quantities
- Trade reasoning and constraint explanations
- Net portfolio impact calculation

6.6 Current Scenario Identification Algorithm

Mathematical Implementation:

```javascript
getCurrentScenario: function(themes) {
    // Current scenario determined by themes > 0.5 (active threshold)
    const currentBinary =
        (themes.usd > 0.5 ? 1 : 0) +        // Bit 0
        (themes.ai > 0.5 ? 2 : 0) +         // Bit 1
        (themes.pe > 0.5 ? 4 : 0) +         // Bit 2
        (themes.intl > 0.5 ? 8 : 0);        // Bit 3

    return currentBinary + 1; // Convert to 1-16 ID range
}
```

Display Requirements:

- Current scenario visually distinct from all others
- Border color: #007bff (blue)

- Border width: 3px
- Box shadow: 0 4px 20px rgba(0,123,255,0.3)
- Scale transform: 1.02
- Label positioning: Absolute, centered above card

---

7. Critical Browser Restrictions

7.1 Storage API Limitations

CRITICAL: localStorage/sessionStorage NOT SUPPORTED in Claude.ai artifacts environment

Exception Handling: If user explicitly requests localStorage usage, explain Claude.ai limitations and offer in-memory alternatives.

```javascript
// ❌ WRONG - Will cause artifacts to fail
localStorage.setItem('data', JSON.stringify(state));
sessionStorage.setItem('temp', value);

// ✅ CORRECT - Use in-memory storage instead
// For React components:
const [state, setState] = useState(initialState);

// For HTML artifacts:
let memoryStorage = {};
function setItem(key, value) { memoryStorage[key] = value; }
function getItem(key) { return memoryStorage[key]; }
```

7.2 Production vs Development Storage

Production Environment (web deployment):

```javascript
```

```
// Full localStorage support available
const stateToSave = {
    version: this.version,
    currentStep: this.currentStep,
    state: this.state
};
localStorage.setItem('hcp_tracker_state', JSON.stringify(stateToSave));
```

Development/Artifact Environment:

```
javascript

// Memory-only storage pattern
const StorageAdapter = {
    memoryStore: {},
    setItem: function(key, value) {
        this.memoryStore[key] = value;
    },
    getItem: function(key) {
        return this.memoryStore[key] || null;
    }
};
```

---

8. Implementation Error Patterns

8.1 Critical Failure Patterns

Pattern 1: Undefined Variable References Symptom: All theme probabilities show uniform values (15.0%) Root Cause: Variable scope issues during module embedding

```
javascript
```

```
// ❌ WRONG - themeData undefined in embedded function
Object.entries(themeIndicators).forEach(([key, config]) => {
    const indicator = themeData[config.dataKey]; // ❌ themeData undefined
});

// ✅ CORRECT - Proper variable definition
const themeData = dataIndicators[themeName]; // ✅ Define before use
Object.entries(themeIndicators).forEach(([key, config]) => {
    const indicator = themeData[config.dataKey];
});
```

Pattern 2: Momentum Calculation Baseline Errors Symptom: All theme probabilities show 50.0% (neutral fallback) Root Cause: Momentum comparison uses wrong historical baseline

javascript

```
// ❌ WRONG - Compare current vs immediate previous (always ≈ same)
const previous = indicator.history[indicator.history.length - 1]; // ❌ Same as current
const momentum = (current - previous) / previous; // ❌ Always ≈ 0

// ✅ CORRECT - Compare current vs 6 periods historical baseline
const previous = indicator.history[indicator.history.length - 6]; // ✅ 6 periods back
const momentum = (current - previous) / previous; // ✅ Shows real momentum
```

Pattern 3: Data Structure Mismatches Symptom: Module functions fail to find indicators Root Cause: Different data key naming between modules

javascript

```
// ❌ WRONG - Inconsistent naming conventions
const indicators = {
    'qqqSpy': data, // camelCase
    'netMargins': data, // camelCase
    'qqq_spy': data, // snake_case - MISMATCH
};

// ✅ CORRECT - Standardize on snake_case for all indicators
const indicators = {
    'qqq_spy': data, // snake_case
    'net_margins': data, // snake_case
    'forward_pe': data, // snake_case
};
```

Pattern 4: Step 4 Display Regressions Symptom: Scenarios display in probability ranking instead of binary order Root Cause: Using probability-sorted array instead of ID-sorted array

```javascript
// ❌ WRONG - Probability ranking display
scenarios.forEach(scenario => {
  // Displays in rank order #1, #2, #3...
});

// ✅ CORRECT - Binary order display
const scenariosInBinaryOrder = [...scenarios].sort((a, b) => a.id - b.id);
scenariosInBinaryOrder.forEach(scenario => {
  // Displays in binary order S1, S2, S3...
});
```

NEW Pattern 5: Rebalancing Calculation Errors - v1.3 Symptom: Incorrect trade recommendations or constraint violations Root Cause: PIMIX/PYLD rules not properly applied

```javascript
// ❌ WRONG - Generate BUY orders for PIMIX
if (driftAnalysis.PIMIX.drift < 0) {
  trades.push({ security: 'PIMIX', action: 'BUY' }); // ❌ Violates hold-only rule
}

// ✅ CORRECT - PIMIX hold-only enforcement
if (driftAnalysis.PIMIX.drift > 0) {
  trades.push({
    security: 'PIMIX',
    action: 'SELL', // ✅ Only SELL allowed
    constraints: ['PIMIX_HOLD_ONLY']
  });
}
// Never generate PIMIX BUY orders
```

8.2 Debug Console Procedures

Theme Analysis Debugging:

```javascript
```

```
// Enable detailed logging for troubleshooting
console.log('=== THEME MOMENTUM CALCULATION ===');
console.log('Data structure themes:', Object.keys(dataIndicators));
console.log('Individual indicator momentum:', momentum);
console.log('Final theme probabilities:', themeProbabilities);

// Validate data structure
Object.entries(dataIndicators).forEach(([theme, themeData]) => {
    console.log(`${theme}:`, Object.keys(themeData));
});

// Check momentum calculations
Object.values(dataIndicators).forEach(theme => {
    Object.values(theme).forEach(indicator => {
        if (indicator.history && indicator.history.length >= 6) {
            const current = indicator.current;
            const baseline = indicator.history[indicator.history.length - 6];
            console.log(`${indicator.name}: current=${current}, 6-back=${baseline}, momentum=${((current-baseline)/baseli
        }
    });
});
```

Step 4 Display Debugging:

javascript

```
// Validate Step 4 display requirements
console.log('=== STEP 4 DISPLAY VALIDATION ===');
console.log('Scenarios count:', scenarios.length);
console.log('Display order check:', scenarios.map(s => s.id));
console.log('Current scenario ID:', getCurrentScenario(themes));
console.log('Color class distribution:', scenarios.map(s => getScenarioColorClass(s.probability)));

// Check DOM elements
const scenarioCards = document.querySelectorAll('.scenario-card');
const currentScenarioCard = document.querySelector('.current-scenario');
console.log(`DOM: ${scenarioCards.length} cards, current scenario highlighted: ${!!currentScenarioCard}`);
```

NEW v1.3 - Rebalancing Debugging:

javascript

```
// Validate rebalancing calculations
console.log('=== REBALANCING CALCULATION VALIDATION ===');
console.log('Current positions:', currentPositions);
console.log('Target allocation:', targetAllocation);
console.log('Drift analysis:', driftAnalysis);
console.log('Generated trades:', trades);

// Check constraint application
trades.forEach(trade => {
    console.log(`${trade.security} ${trade.action}: $${trade.dollarAmount} - ${trade.reason}`);
    if (trade.constraints) {
        console.log(`  Constraints applied: ${trade.constraints.join(', ')}`);
    }
});

// Validate PIMIX/PYLD rules
const pimixBuys = trades.filter(t => t.security === 'PIMIX' && t.action === 'BUY');
if (pimixBuys.length > 0) {
    console.error('❌ CRITICAL ERROR: PIMIX BUY orders generated (violates hold-only rule)');
}

const incomeBuys = trades.filter(t => ['PIMIX', 'PYLD'].includes(t.security) && t.action === 'BUY');
const nonPyldIncomeBuys = incomeBuys.filter(t => t.security !== 'PYLD');
if (nonPyldIncomeBuys.length > 0) {
    console.error('❌ ERROR: Income BUY orders not routed to PYLD');
}
```

Error Detection:

```
javascript
```

```
// Monitor for common calculation failures
if (Object.values(themeProbabilities).every(p => Math.abs(p - 0.15) < 0.001)) {
    console.error('❌ All themes showing 15% - indicates data validation failure');
}

if (Object.values(themeProbabilities).every(p => Math.abs(p - 0.50) < 0.001)) {
    console.error('❌ All themes showing 50% - indicates momentum calculation failure');
}

if (Object.values(themeProbabilities).some(p => isNaN(p))) {
    console.error('❌ NaN values in theme probabilities - indicates calculation error');
}

// Step 4 display validation
const scenarioCards = document.querySelectorAll('.scenario-card');
if (scenarioCards.length !== 16) {
    console.error('❌ Step 4: Wrong number of scenario cards displayed');
}

const firstScenarioId = document.querySelector('.scenario-card .scenario-id')?.textContent;
if (firstScenarioId !== 'S1') {
    console.error('❌ Step 4: Scenarios not in binary order - shows probability ranking');
}

const currentScenario = document.querySelector('.current-scenario');
if (!currentScenario) {
    console.error('❌ Step 4: Current scenario highlighting missing');
}

// NEW v1.3 - Rebalancing validation
if (typeof RebalancingModule === 'undefined') {
    console.error('❌ RebalancingModule not loaded');
}

const totalDrift = Object.values(driftAnalysis || {}).reduce((sum, d) => sum + Math.abs(d.drift || 0), 0);
if (totalDrift > 2.0) { // More than 200% total drift indicates calculation error
    console.error('❌ Excessive portfolio drift calculated - check position inputs');
}
```

9. Browser Compatibility

9.1 Minimum Requirements

JavaScript Features Required:

- ES6+ support (const, let, arrow functions, template literals)
- JSON.parse/stringify with error handling
- localStorage with quota management
- HTML5 file input APIs
- NEW v1.3: Number input validation for position entry

Tested Browser Matrix:

- Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
- Mobile: iOS Safari 14+, Chrome Mobile 90+

9.2 Fallback Strategies

localStorage Unavailable:

- Graceful degradation to session-only state
- Warning message to user about persistence loss

File API Unavailable:

- Sample data generation remains functional
- Manual data entry as alternative

NEW v1.3 - Input Validation Fallbacks:

- Server-side validation patterns for older browsers
- Progressive enhancement for advanced input types

---

10. Security Considerations

10.1 Data Privacy

No External Communications: All processing occurs locally in browser
No Persistent Tracking: Only functional localStorage for user convenience

No Data Transmission: User data never leaves local environment **NEW v1.3:** Position data remains local and is not transmitted

10.2 Input Validation

File Upload Security:

- JSON parsing with try/catch error handling
- Schema validation for expected data structure
- Size limits on uploaded files
- Sanitization of user inputs in manual overrides

NEW v1.3 - Position Input Security:

- Number validation for share quantities and dollar amounts
- Range checks for reasonable portfolio values
- Sanitization of position descriptions and notes

---

11. Testing Framework

11.1 Unit Testing Strategy

Module-Level Testing:

```
javascript
```

```
// Example test structure
testThemeCalculatorV210() {
    const testData = FileHandler.generateSampleData('tech_boom');
    const analysis = ThemeCalculator.calculateThemeAnalysis(testData);
    assert(analysis.themes.ai > 0.70, 'AI theme should be strong in tech boom');
    assert(analysis.scenarios.length === 16, 'Must generate all 16 scenarios');
    assert(Math.abs(totalProbability - 1.0) < 0.01, 'Scenarios must sum to 100%');

    // Step 4 display tests
    const currentScenario = ThemeCalculator.getCurrentScenario(analysis.themes);
    assert(currentScenario >= 1 && currentScenario <= 16, 'Current scenario ID valid');

    const colorClass = ThemeCalculator.getScenarioColorClass(0.30);
    assert(colorClass === 'scenario-very-high', 'Color class correct for 30% probability');
}

// NEW v1.3 - Rebalancing tests
testRebalancingModule() {
    const currentPositions = { VTI: {shares: 100, value: 25000, percentage: 0.50} };
    const targetAllocation = { VTI: 0.35, VEA: 0.20, /* ... */ };
    const totalValue = 50000;

    const drift = RebalancingModule.calculateDrift(currentPositions, targetAllocation, totalValue);
    assert(drift.VTI.drift > 0.10, 'VTI should show overweight drift');

    const trades = RebalancingModule.generateTrades(drift, currentPrices, constraints);
    assert(trades.some(t => t.security === 'VTI' && t.action === 'SELL'), 'Should generate VTI sell');
    assert(!trades.some(t => t.security === 'PIMIX' && t.action === 'BUY'), 'No PIMIX buys allowed');
}
```

Integration Testing:

- File data flow: FileHandler → TrackerCore → ThemeCalculator → PortfolioOptimizer → RebalancingModule → Display
- State persistence: Save → Reload → Verify consistency
- Error recovery: Invalid data → Graceful handling → User notification

11.2 Validation Scenarios

Required Test Cases:

1. All 5 sample scenarios generate expected probability ranges

2. Manual overrides properly update calculations
3. Navigation validation prevents invalid step advancement
4. State persistence survives browser refresh
5. File upload handles malformed JSON gracefully
6. **Step 4 scenarios display in binary order**
7. **Current scenario highlighting functions correctly**
8. **Probability color coding matches specification**
9. **NEW v1.3: Position input validates and calculates drift correctly**
10. **NEW v1.3: Trade generation respects PIMIX/PYLD constraints**
11. **NEW v1.3: 🔄 Position limits not enforced (under consideration)**

Expected Result Ranges (Critical for Validation):

- **Tech Boom Scenario:** AI 70-85%, USD 20-35%, P/E 30-45%, International 25-40%
- **USD Strength Scenario:** USD 65-80%, AI 15-30%, P/E 20-35%, International 10-25%
- **P/E Reversion Scenario:** P/E 70-85%, USD 30-45%, AI 25-40%, International 20-35%
- **International Scenario:** International 70-85%, USD 15-30%, AI 20-35%, P/E 25-40%

NEW v1.3 - Rebalancing Test Scenarios:

- **PIMIX Hold-Only:** Never generate PIMIX BUY orders under any circumstances
- **PYLD Primary Income:** Route all income increases through PYLD
- **Drift Threshold:** Ignore drifts <1% unless cash would go negative
- 📋 **Constraint Testing:** Tax optimization and order types out of scope

---

12. Data Collector Integration

12.1 File Format Specifications

Expected JSON Structure:

```javascript
```

```
{
  "version": "3.8+",
  "type": "monthly|initialization",
  "timestamp": "ISO_8601_string",
  "indicators": {
    // Nested theme structure as documented in FileHandler v1.5
  },
  "trading_status": "GREEN|YELLOW|RED"
}
```

12.2 Version Compatibility

Backward Compatibility: Support Data Collector v3.6+ output formats

Forward Compatibility: Graceful handling of unknown fields in newer versions

Error Recovery: Clear messaging when file format unsupported

---

13. Deployment Architecture

13.1 Single-File Strategy

Embedded Module Pattern:

```
html

<script>
// Module definitions embedded directly
const TrackerCore = { version: '1.2', /* implementation */ };
const FileHandler = { version: '1.5', /* implementation */ };
const ThemeCalculator = { version: '2.10', /* implementation */ };
const PortfolioOptimizer = { version: '2.0', /* implementation */ };
const RebalancingModule = { version: '1.0', /* NEW v1.3 implementation */ };
// etc.
</script>
```

Advantages:

- Zero external dependencies
- Offline functionality
- Simple deployment (single file)
- No CORS issues

- No version synchronization problems

13.2 Development vs Production

Development Mode:

- External module files for easier editing
- Detailed console logging enabled
- Integration test harness available

Production Mode:

- Embedded modules for deployment
- Error logging minimized
- Optimized file size

---

14. Extension Points

14.1 Future Module Integration

Designed Extension Areas:

- Steps 8-10: New modules can hook into existing workflow
- Alternative optimization engines: Modular replacement of portfolio optimization
- Data source adapters: Additional data collector formats
- Export formatters: Multiple output format support
- 📋 **Tax Optimization Module:** Framework preserved for future implementation
- 📋 **Advanced Trade Execution:** Order management system for future enhancement

14.2 API Readiness

Module Interfaces:
Each module exposes standardized interface:

```javascript
```

```
ModuleName: {
  version: string,
  calculateXXX: function(data) → result,
  displayXXX: function(result, containerId),
  validateXXX: function(data) → validation
}
```

NEW v1.3 - Rebalancing Interface:

```
javascript

RebalancingModule: {
  version: '1.0',
  calculateDrift: function(positions, targets, totalValue) → driftAnalysis,
  generateTrades: function(drift, prices, constraints) → tradeList,
  displayPositions: function(positions, containerId),
  displayTrades: function(trades, containerId),
  validatePositions: function(positions) → validation
}
```

---

15. Change Management

15.1 Version Strategy

Module Independence: Each module maintains separate version numbers

Compatibility Matrix: Document module version compatibility in PRD

Release Coordination: Major releases coordinate compatible module versions

15.2 Update Procedures

Development Updates:

1. Test new module version in isolation
2. Integration test with current module suite
3. Update compatibility matrix
4. Embed in production HTML

Breaking Changes:

- Major version increment required
- Migration guide provided

- Backward compatibility period where possible

NEW v1.3 - Rebalancing Update Procedures:

- Test PIMIX/PYLD constraint enforcement
- Validate drift calculations across scenarios
- Verify trade generation logic
- 🔄 **Position limit enforcement under consideration:** Test with and without limits

---

End of Technical Specification v1.3 - Enhanced with rebalancing implementation specifications, preserved all existing functionality with clear scope annotations for Steps 6-7 development