**Comprehensive Strategy Calculators Project Plan**

**Overall Objective**

Create a complete, high-performance strategy\_calculators.py file with 500 football bet type calculation methods that are accurate, well-documented, optimized, and production-ready for the Heage Betting App.

**Breaking Down the Task**

Given Claude's context window limitations and the scope of the project (500 bet types, approximately 20,000 lines of code), we'll divide the work as follows:

1. **5 Implementation Claudes**: Each implementing 100 bet types
2. **1 Manager Claude**: Overseeing the project, integrating code, handling reviews
3. **Review Support**: Grok and ChatGPT for cross-checking

**Team Structure**

**Implementation Claudes**

* **Claude 1**: Setup preamble + Bet Types 1-100 (Match Outcome & Goals Part 1)
* **Claude 2**: Bet Types 101-200 (Goals Part 2 & Score)
* **Claude 3**: Bet Types 201-300 (Player-Specific)
* **Claude 4**: Bet Types 301-400 (Team Event)
* **Claude 5**: Bet Types 401-500 (Game Incident & Specialty/Combo)

**Manager Claude**

* Oversees implementation
* Provides feedback to Implementation Claudes
* Integrates chunks into master file
* Coordinates with Grok & ChatGPT for review
* Ensures consistency across the codebase

**Implementation Approach**

**Architecture Principles**

1. **Consistency**: All bet type methods must follow the same structure and parameter patterns
2. **Performance**: Use caching for repeated calculations, optimize multi-outcome processing
3. **Robustness**: Comprehensive validation, precise error handling, Decimal for financial calculations
4. **Maintainability**: Well-documented code, clear logging, consistent naming
5. **Testability**: Each method returns standardized output, handles edge cases

**Code Standards**

1. **Method Signatures**: All follow calculate\_bet\_type(back\_odds, lay\_odds, stake, commission=0.02, back\_commission=0.00) pattern
2. **Output Format**: All methods return consistent dictionary via create\_standard\_result()
3. **Error Handling**: All methods validate inputs and handle edge cases identically
4. **Documentation**: All methods include descriptive docstrings with examples
5. **Logging**: Consistent logging of calculation inputs, results, warnings, and errors

**Implementation Process**

**Phase 1: Initial Implementation**

Each Implementation Claude creates their assigned chunk following the provided template, with tests and documentation.

**Phase 2: Review & Refinement**

Manager Claude coordinates review with Grok and ChatGPT, then returns feedback to Implementation Claudes for refinement.

**Phase 3: Integration**

Manager Claude combines all approved chunks into the final file, ensuring consistency across the full codebase.

**Phase 4: Final Verification**

Complete testing of the full integrated file, with a focus on edge cases and integration points.

**Timeline**

1. Initial Implementation: 1-2 hours per Claude
2. Review & Refinement: 1 hour per chunk
3. Integration: 1 hour
4. Final Verification: 1 hour

Total estimated time: 10-15 hours of work, can be completed within 1-2 days with parallel processing.