Technical Approach: Betting Extension Development

# Core Architecture

Extension Structure

* - Chrome Extension Manifest V3
* - Content script for DOM interaction
* - Background service worker for state management
* - Popup UI and dedicated pages for match/bet display

Data Flow

* - Content script → Background service worker → Storage → UI
* - Real-time updates via Chrome messaging API
* - Persistent storage using Chrome Storage API

# Technical Implementation

Match Selection Detection

* - MutationObserver to track DOM changes on betting site
* - CSS selector targeting for match elements and odds extraction
* - Event delegation for efficient DOM event handling

Betting Algorithm

* - Binary classification of selections (favorites vs. underdogs)
* - Weighted random selection maintaining 60/40 ratio
* - Hash-based tracking of previous combinations
* - Single bet placement with randomized player selection

State Management

* - Centralized state in background service worker
* - Chrome Storage API for persistent data across page refreshes
* - Message-based communication between components
* - Event-driven architecture for real-time updates
* - Data persistence for selected matches and bet history when page refreshes

# UI Components

Popup Interface

* - Minimalist control panel with start/stop functionality
* - Match count display with selection summary
* - Navigation to detailed views

Match Display Page

* - Responsive grid layout for selected matches
* - Visual indicators for favorites/underdogs
* - Real-time updates when selections change

Bet History Page

* - Chronological list of placed bets
* - Detailed view of selection combinations
* - Statistics on favorite/underdog distribution

# Error Handling & Resilience

DOM Structure Changes

* - Fallback selectors for critical elements
* - Graceful degradation when elements can't be found
* - Automatic recovery mechanisms

Extension Context Issues

* - State preservation during context invalidation
* - Recovery procedures for interrupted operations
* - User notifications for critical failures

# Development Timeline

Day 1: Core functionality implementation

* - Extension architecture setup
* - Match selection detection system
* - Betting algorithm implementation

Day 2: UI development and testing

* - User interface components
* - Automated betting system
* - Testing and refinement

# Technical Challenges & Solutions

DOM Reliability

* - Challenge: Betting site may change structure
* - Solution: Multiple selector strategies with fallbacks

Selection Algorithm

* - Challenge: Maintaining 60/40 ratio with limited selections
* - Solution: Adaptive weighting based on available options

Performance

* - Challenge: Efficient DOM observation without overhead
* - Solution: Targeted mutation observation with event throttling

Data Persistence

* - Challenge: Maintaining state across page refreshes and browser restarts
* - Solution: Chrome Storage API with background service worker state management

# Deliverables

Chrome Extension Package

* - Complete extension with all required functionality
* - Ready for installation in Chrome browser
* - Compatible with target betting site

Core Features

* - Match selection tracking system
* - 60/40 favorite/underdog betting algorithm
* - Duplicate combination prevention
* - Bet history for a session

Documentation

* - Installation instructions
* - Usage guide
* - Technical overview for future maintenance

# Success Criteria

Extension correctly tracks selected matches on the betting site

Betting algorithm maintains 60/40 ratio of favorites to underdogs

System prevents duplicate bet combinations

UI clearly displays selected matches and bet history

Automated betting functions reliably with start/stop controls

Selected matches and bet history persist across page refreshes and browser restarts

# Conclusion

This technical approach leverages modern browser extension capabilities to create a robust betting automation system. By using an event-driven architecture with centralized state management, we ensure reliable operation and data consistency. The implementation focuses on core functionality while building in error resilience to handle the unpredictable nature of web DOM interactions.

The 2-day development timeline is achievable by focusing on essential technical components first, followed by UI implementation and testing. This approach ensures delivery of a functional extension that meets all key requirements while maintaining code quality and performance.