He Maumahara System Overview

Version: v3.0.1
Date: 2026-01-15
Status: Stable Release

1. Introduction

He Maumahara (“The Memory”) is a browser-based memory game designed for culturally grounded, bilingual learning and cognitive engagement. The experience targets accessibility for Kaumātua and whānau (high contrast UI, large touch targets, reduced clutter), while delivering a technically robust and privacy-preserving adaptive system.

The system’s differentiator is its fully client-side personalization loop:

- Gameplay telemetry is stored locally (IndexedDB).
- A **Flow Index** (0.0–1.0) summarizes “how well the round went” using a lightweight fuzzy-logic system.
- A **contextual bandit** (LinUCB) chooses the next difficulty configuration (three discrete arms).
- The next configuration is persisted locally (localStorage) and applied on the next run.

No backend is required. No gameplay data is uploaded.

2. Design Philosophy

2.1 Cultural Safety and Integration

The project is built to support culturally resonant engagement rather than generic “gamification”.

- Māori language and imagery are treated as core product value rather than decoration.
- The UI and pacing are designed to be supportive and non-punitive.
- The system avoids ranking/leaderboard mechanics; the primary feedback signal is Flow.

2.2 Privacy-First Architecture (Local-Only by Default)

The architecture is “zero data exfiltration”:

- All computation (gameplay logic, analytics, AI adaptation) runs in the browser.
- Storage is device-local: IndexedDB and localStorage.
- Export is user-initiated (JSON download) for evaluation and reproducibility.

2.3 Flow Theory as the Core Objective

The system optimizes for sustained engagement by balancing challenge with skill:

- **Flow Index** is the primary post-game metric.
- The adaptation goal is not simply “maximize difficulty” or “minimize time”, but to keep challenge appropriate over time.

2.4 Accessibility and Cognitive Load Management

Key choices reflect senior-friendly design:

- High contrast colors and large typography.
- Large card targets and consistent spacing.
- Minimal navigation and low UI noise during play.

3. Product Surface: Pages and User Journey

3.1 Pages

- index.html: home menu (Play, Analytics, Instructions, Credits)
- play.html: level selection
- lvl-1.html, lvl-2.html, lvl-3.html: gameplay pages
- analytics.html: analytics dashboard (history + demo)
- instructions.html, credits.html: informational pages

3.2 Typical Journey

1. Home → Level selection → Play a level
2. Game-over screen shows:
    - elapsed time and post-game summary panel
    - Flow Index feedback
    - export/screenshot options
    - “Next Game” progression using the next AI configuration
3. Analytics page displays session history and overall trends.

4. Core Features (User-Visible)

4.1 Multi-Level Cognitive Tasks

- **Level 1 (Visual baseline)**: image–image matching with a stable, onboarding-friendly layout.
- **Level 2 (Spatial challenge)**: variable layouts and adjacency-based placement to tune search complexity.
- **Level 3 (Linguistic challenge)**: image–text matching (kupu), adding semantic recall on top of visual memory.

4.2 Shared Gameplay Controls

Gameplay pages include a consistent control set:

- Show: temporary reveal of cards with a time cost (hint usage)
- Export: telemetry download (JSON) for reproducibility
- Adapt: toggle adaptive difficulty (persisted)
- Reset: clear per-level telemetry store

4.3 Analytics and Progress Feedback

- Post-game summary panel on each level page.
- Analytics dashboard with:
    - session history (IndexedDB game_history)
    - an overall review widget based on local clustering (K-Means).

5. System Architecture (Implementation View)

5.1 Layers

- **Presentation**: static HTML + CSS
- **Gameplay logic**: vanilla JavaScript per level
- **AI and analytics**: vanilla JavaScript modules
- **Persistence**:
    - IndexedDB for telemetry, game history, and AI profile state
    - localStorage for small settings and next-round configuration

5.2 Key Modules

- js/game-core.js: shared utilities, telemetry helpers, export/screenshot integration
- js/lvl1.js, js/lvl2.js, js/lvl3.js: level-specific gameplay logic
- js/ai-helper.js: extracts metrics from telemetry and orchestrates AI at game end
- js/ai-engine.js: Flow Index scoring, contextual bandit decision layer, config generation
- js/analytics-summary.js: post-game summary UI and overall review rendering
- js/game-history.js: game_history storage access

6. AI: Adaptive Difficulty Engine (Detailed Overview)

6.1 Inputs (Telemetry-Derived Metrics)

Common signals used by the AI include:

- completionTime (seconds)
- totalPairs
- totalMatches and failedMatches (error rate)
- totalClicks (click efficiency)
- flipIntervals (cadence stability)
- cheatCount (Show usage)

- consecutiveErrors and maxConsecutiveErrors (difficulty/frustration indicators)

6.2 Flow Index (Fuzzy Logic)

The Flow Index compresses multi-dimensional gameplay signals into a stable scalar:

- Designed for interpretability and robustness to noisy inputs.
- Supports a user-facing display layer that can be clamped to avoid discouraging feedback.

6.3 Decision Layer (Contextual Bandit: LinUCB)

The bandit selects among three discrete “arms”:

- Arm 0: easiest
- Arm 1: standard
- Arm 2: challenge

The reward signal is derived from the Flow Index, enabling online learning per device without centralized training.

6.4 Configuration Outputs

Parameters adapted across rounds include:

- grid size (notably Level 2 and 3: 5×4 vs 4×6)
- reveal dynamics (hideDelay and showScale)
- Level 2 adjacency assistance targets
- step smoothing to prevent abrupt difficulty jumps

7. Data Model (Storage and Ownership)

7.1 IndexedDB Stores (Conceptual)

- telemetry_lvl1 / telemetry_lvl2 / telemetry_lvl3: per-level event streams
- game_history: completed sessions for the analytics dashboard
- ai_player_profile: persisted AI player profile and bandit state

7.2 localStorage Keys (Conceptual)

- ai_adaptive_enabled
- ai_level1_config / ai_level2_config / ai_level3_config

8. Testing, Evaluation, and Demonstration

- Automated player simulations exist under tests/ to regression-test gameplay and AI behavior.
- Exported telemetry streams enable offline inspection and reproducible analysis during evaluation.

9. Limitations and Future Work

9.1 Known Constraints

- Device-local progress: no cross-device syncing by default.
- Cold start: adaptation stabilizes after a small number of rounds.
- Browser storage limits: very long-term logging may reach quota on some devices.

9.2 Roadmap Options

- Manual export/import of profiles to support device migration.
- Caregiver-friendly analytics mode.
- Extended linguistic tasks and/or voice features for Level 3.