Implementation Plan: Nodges Refactoring & Stabilization

Goal

Stabilize the Nodges codebase by enforcing strict TypeScript usage, removing legacy data handling, and resolving all linting errors/warnings (specifically `@ts-ignore` usages).

Prioritized Tasks

Phase 1: Core Type Safety (App.ts & Imports)

Target: `src/App.ts`, `src/core/*.ts`

- [ ] **Fix Imports:** Ensure all managers are imported correctly. If a Manager is still `.js`, generate a `.d.ts` declaration file or migrate it to `.ts`.
- [ ] **Remove `@ts-ignore`:** SYSTEMATICALLY go through `App.ts`.
  - For `SelectionManager`, `RaycastManager`, etc., create proper Interfaces/Types.
  - Inject explicit types instead of `any` where possible.
- [ ] **Standardize `currentGraphData` Access:** Ensure `App.ts` *only* uses `this.currentEntities` and `this.currentRelationships`.

Phase 2: JavaScript to TypeScript Migration

Target: `src/effects/`, `src/utils/`

- [ ] **Migrate `HighlightManager.js` to `.ts`:**
  - This is a critical system. It needs strict types for `highlightRegistry` and `userData`.
  - Define interfaces for `HighlightData` and `HighlightOptions`.
- [ ] **Migrate `SelectionManager.js` to `.ts`:**
  - Type the selection events and state updates.

Phase 3: Edge & Node Manager Alignment

Target: `src/core/EdgeObjectsManager.ts`

- [ ] **Strict Typing for Edges:**
  - Update `updateEdges` signature to explicitly accept `RelationshipData[]`.
  - Remove `any` casts inside the loop.
  - Ensure `userData` in generated Meshes matches a defined Interface (`EdgeUserData`).

Verification Plan

Automated

1. Run `npm run type-check` (or `tsc --noEmit`) to verify zero errors.
2. Ensure build process completes without warnings.

Manual

1. **Load Test:** Load `openai_10.json` and `nervous_system_full.json`.
2. **Interaction Test:**
   - Hover Nodes (check Highlight color).
   - Hover Edges (check Outline).
   - Click Node (Selection Glow).
   - Change Visual Mappings (UI Panel) -> Verify Scene Updates.