\= SPEC-2: Gamebook Editor – Node Title & Button Enhancements \:sectnums: \:toc:

\== Background

SPEC-1 delivered a functional drag-and-drop web editor and mobile shell for purely branching narrative (no RPG mechanics). Authors can create scene nodes with markdown description, optional image, and action buttons leading to other nodes. During testing, writers highlighted three pain points:

- Absence of a dedicated **Title** field per scene makes large stories hard to navigate in the graph view.
- IDs are implicit (array index) and limited to two digits – this blocks stories > 100 nodes.
- Updating button counts/paths is cumbersome; each change forces manual JSON edits.

SPEC-2 removes those frictions while preserving all existing functionality.

\== Requirements

Requirements follow MoSCoW prioritisation.

Must

- **Node IDs** – Every node has a unique, *editable* positive integer ID (`1, 2, 3…`) with no upper limit. ID collisions are prevented at save time. The UI may optionally display leading zeros, but storage is plain integer.
- **Title Field** – A new required text field stored alongside the description; existing scenes migrate with *blank* titles so authors can curate manually.
- **Description Field** – Keep existing markdown description area.
- **Image** – Upload & preview stored in the existing **Supabase Storage bucket** (`storycraft-images`). Allow JPG/PNG/WebP up to **20 MB**; no server-side thumbnail generation – the editor scales images client-side for graph view.
- **Buttons (1-4)** – UI control to choose button count (min 1, max 4). Each button stores `label` and `targetNodeId`.
- **Graph Consistency** – Editor warns if a button references a non-existing node or creates an orphan.

Should

- Inline validation messages (e.g., "ID already used").
- Keyboard shortcuts: ⌘+S save, ⌘+F jump to node.
- Undo/redo stack (10 steps).

Could

- Dark/light themes with Tailwind + DaisyUI.
- Search bar filtering by ID or Title.
- Node mini-map overview panel.

Won’t

- RPG stats, dice mechanics (out of scope until SPEC-3).

\== Method

\=== 1. Architecture Overview

```
@startuml
skinparam packageStyle rect
package "Front-end (React)" {
  [Graph Canvas]
  [Node Editor Modal]
  [ImageUploader]
}
package "Supabase" {
  [Postgres DB] as DB
  [Storage Bucket] as Bucket
  [Edge Functions] as Fn
}
[Graph Canvas] --> [Node Editor Modal]
[Node Editor Modal] --> Fn : REST RPC
Fn --> DB : CRUD nodes / buttons
ImageUploader --> Bucket : upload
@enduml
```

\=== 2. Supabase Postgres Schema

```
-- nodes table
create table public.nodes (
    id          integer primary key generated always as identity,
    title       text    not null,
    description text    default '',
    image_url   text,
    created_at  timestamptz default now(),
    updated_at  timestamptz default now()
);

-- buttons table (1-4 per node)
create table public.buttons (
    id              bigint primary key generated always as identity,
    node_id         integer not null references public.nodes(id) on delete
cascade,
    idx             smallint not null check (idx between 1 and 4), -- order on
screen
```

```
    label           text    not null,
    target_node_id  integer not null references public.nodes(id),
    unique(node_id, idx)
);
```

Row-level Security (RLS) policies will restrict access to story owners (future multi-author support).

\=== 3. Edge Functions (TypeScript)

- `upsert_node(payload)` – Insert or update node + associated buttons in a single transaction; rejects duplicate IDs.
- `validate_graph()` – Returns list of dangling or orphaned nodes; called after save.
- `search_nodes(query)` – ID/Title full-text search (used by ⌘+F).

\=== 4. Image Workflow

1. Author selects local file ≲ 20 MB.
2. `ImageUploader` component streams file to `storycraft-images` bucket under path `node/{nodeId}/{uuid}.ext`.
3. Stored URL is written to `nodes.image_url` and returned to the editor for immediate preview. Graph view uses CSS background-size to display a scaled image when zoom ≥ 150%.

\=== 5. Front-end Changes

- **Node Editor Modal** gains Title input and Button count selector (1-4). Buttons render dynamically with label + target ID dropdown (populated from nodes query).
- **Graph Canvas** (Cytoscape) labels each node as `ID · Title` and displays a CSS-scaled background image when zoomed in.
- **Validation overlay**: on save, graph highlights invalid edges in red and lists errors.
- **Auto-layout** keeps node positions unchanged to preserve user layout.

\=== 6. Risk & Mitigations

Risk	Impact	Mitigation
Large uploads stall UI	Medium	Use `fetch` with progress; show cancel button
ID collision after manual edit	High	Unique constraint + pre-save check
Orphan nodes after mass delete	Low	`validate_graph()` on each save

\== Implementation

Implementation is divided into four incremental phases so new functionality can be tested and merged without blocking authors.

\=== Phase 0 – Supabase Setup (½ day)

- Provision **storycraft** Supabase project.
- Create `nodes`, `buttons` tables and `storycraft-images` bucket.
- Add RLS policies (owner_id placeholder).
- Configure environment variables in existing Node.js server (service key, anon key, URL).

\=== Phase 1 – Edge Functions & SDK (1½ days)

- Scaffold `upsert_node`, `validate_graph`, `search_nodes` functions with Supabase CLI.
- Integrate Supabase JS v2 in server and React app.
- Unit-test CRUD operations with PostgREST policies.

\=== Phase 2 – UI Enhancements (2 days)

- Extend Node Editor Modal to include **Title**, image uploader, button count selector, target dropdown.
- Refactor state to store integer IDs (zero-pad only in render).
- Add inline validation messages.

\=== Phase 3 – Graph & UX Polish (1 day)

- Update Cytoscape node renderer to show `ID · Title` and scaled image.
- Implement ⌘+S, ⌘+F, undo/redo stack.
- Add orphan & dangling edge highlighting.

\=== Phase 4 – QA & Release (½ day)

- Smoke-test on desktop and mobile shells.
- Prepare release notes and upgrade doc (new projects only).

\== Milestones

Date	Deliverable
T + 0.5 d	Supabase project live; schema & storage bucket created
T + 2 d	Edge Functions deployed; API integrated into dev branch
T + 4 d	UI enhancements merged; basic create/edit working
T + 5 d	Graph view + validations working; feature-freeze
T + 5.5 d	QA sign-off; v2.0.0 tag released

\== Gathering Results

After first week of production use gather:

- Number of nodes edited per story (benchmark vs SPEC-1).
- Time-to-create for a 50-scene branching chapter (author survey).

- Error logs for ID collisions, orphan detection.

Success is declared if **author creation speed improves by $\geq$ 30 %** and no critical errors are reported in first month.