Below is a conceptual **graph-based data model** that captures:

1. **Zymes as first-class nodes**
2. **Internal levels (L0–L3) housed in each node**
3. **Cross-links ("hops") between any two nodes** (keywords, people, places, concepts…)
4. **Breadcrumb-style parent-child hierarchy** within a single Zyme
5. **Arbitrary "related" edges** between Zymes

---

1 · Core Entity: ZymeNode

Each Zyme lives as a single node in your knowledge graph.

```
ZymeNode {
  id: string,              // unique URI or UUID
  title: string,           // document title or term name
  type: string,            // e.g. "legal_opinion","concept","person","event"
  metadata: {              // provenance, authors, date, etc.
    source_url?: string,
    authors?: [string],
    pub_date?: string,
    description?: string
  },
  levels: {                 // the playbook L0–L3 bundle
    L0: [ HighlightCard ],  // as previously defined
    L1: [ CleanBulletCard ],
    L2: [ SummaryCard ],
    L3: [ SourceBlock ]
  },
  edges: [ Edge ]          // all outward edges from this node
}
```

1.1 · HighlightCard (L0)

```
HighlightCard {
  id: string,              // unique within the ZymeNode
  text: string,            // the bullet
  hero: HeroVisual,        // KPI chart, sparkline, or illustration
  keywords: [string],      // terms that spawn edges if clicked
}
```

And similarly for `CleanBulletCard`, `SummaryCard`, `SourceBlock` (mirroring the JSON schema we defined).

---

2 · Edges

We encode two main edge types:

```
Edge {
  target: string,           // ZymeNode.id of the other node
  relation: string,         // e.g. "CHILD_OF", "RELATED_TO", "DEFINED_BY", "AUTHORED_BY"
  via: {                    // optional context
    level: "L0" | "L1" | …, // which card spawned this edge
    keyword?: string,       // which word was clicked
    hopDistance?: integer   // for graph algorithms
  }
}
```

relation	semantics
`CHILD_OF`	navigated "down" into this node from a parent Zyme
`PARENT_OF`	inverse of `CHILD_OF`
`RELATED_TO`	lateral jump (e.g. two concepts that share context)
`DEFINED_BY`	points to a glossary or definition ZymeNode
`AUTHORED_BY`	links to a Person node representing the author
`EVENT_AT`	links to a Place or Conference node

Hop distance can be precomputed for UX shortcuts (“this term is 3 hops away”).

---

3 · Example Snippet

```
{
  "id": "zyme://doe2025-pnas-summary",
  "title": "P vs. NP — Cook–Levin Theorem",
  "type": "concept",
  "metadata": {
    "source_url": "https://example.com/opinion.pdf",
    "authors": ["Stephen A. Cook","Leonid Levin"],
    "pub_date": "2025-05-01"
  },
  "levels": {
    "L0": [
      {
        "id": "h1",
        "text": "Cook–Levin establishes NP-completeness of SAT.",
        "hero": { "type":"illustration", "ref":"cook-levin.svg" },
        "keywords": ["NP-completeness","SAT"]
      }
    ],
    "L1": [ /* … */ ],
    "L2": [ /* … */ ],
    "L3": [ /* … */ ]
  },
  "edges": [
    {
      "target": "zyme://concept-NP-completeness",
      "relation": "RELATED_TO",
      "via": { "level":"L0", "keyword":"NP-completeness" }
    },
    {
      "target": "zyme://person-stephen-cook",
      "relation": "AUTHORED_BY"
    },
    {
      "target": "zyme://legal-opinion-roe-v-wade",
      "relation": "RELATED_TO",
      "via": { "hopDistance": 3 }
    }
  ]
}
```

---

4 · Graph Storage Options

1. **Property Graph** (Neo4j, Amazon Neptune)
   - Native support for nodes + richly-typed edges, traversals by hop count.
2. **RDF / Triple Store** (GraphDB, Blazegraph)
   - URIs for nodes/relations, SPARQL querying.
3. **Document+Edge Store Hybrid**
   - Store `ZymeNode` JSON in a document DB (MongoDB, Elasticsearch)
   - Maintain an edge list in a graph engine or adjacency tables in SQL.

---

5 · UI Implications

- **Breadcrumb A** (“Zyme Level”): Shows the current L-level path within a single Zyme node,
  - e.g. `L0 → L1 → L2 → L3`.
- **Breadcrumb B** (“Graph Navigation”): Tracks the **origin Zyme node and the chain of child/related nodes** the user has jumped through,
  - e.g. `Patent Brief (root) → NP-Completeness → Cook–Levin Theorem`.
  - *Note:* the “root” here is the Zyme node they started on, not the global graph root.
- **Global Graph View**: A “My Zyme Map” canvas that visualizes all the Zyme nodes (and edges) the user has visited in this session.
- **Future (advanced)**
  - **Hop-indicator**: could surface “n hops to [Target Zyme]” in an advanced panel or utility tab—deferred for later.

Appendix

A. Pick Your Storage Model

Pick your storage model and map our JSON schema → DB schema.

This refers to taking your working **Zyme output format (JSON)** and connecting it to a **persistent database structure**, so that Zymes, their layers (L0–L3), and graph relationships can be:

- Stored
- Queried
- Rendered dynamically in the UI
- Navigated as a graph (nodes + edges)

This is the **technical decision** about what kind of database architecture best supports Zyme data and its graph-like structure.

Storage Model	Best For…	Examples
Relational (SQL)	Tabular data, strict typing, joins	PostgreSQL (with `jsonb`, GIN)
Document store	Flexible nested data, fast ingest	MongoDB, Firestore
Graph database	Native graph traversal, hop-aware queries	Neo4j, Amazon Neptune
Hybrid	Use SQL/Mongo for Zyme content + GraphDB for edges	SQL + Neo4j combo

---

B. Map JSON Schema → DB Schema

This is where you take the **ZymeNode object** (with L0–L3, glossary, nav, edges) and decide:

a) Where each part lives in your database

JSON Field	DB Target Table	Notes
`id`, `title`, `type`	`zyme`	Already defined in your `timezyme_enhanced_schema.sql`
`levels.L0`, `L1`, …	`zyme_section`	One row per card/section per level
`glossary`	`article_keyword` or new `glossary_term` table	Include source + confidence
`nav.order`	`section_order` field	Already included
`edges`	⚠️ **Needs a new `zyme_edge` table**	To support node-to-node relations

b) Define new tables for graph traversal

```
CREATE TABLE zyme_edge (
    from_id     TEXT,  -- zyme_node ID
    to_id       TEXT,  -- target Zyme or keyword node
    relation    TEXT,  -- 'CHILD_OF', 'DEFINED_BY', etc.
    via_level   TEXT,  -- 'L0', 'L1', ...
    keyword     TEXT,
    hop_distance INT,
    created_at  TIMESTAMPTZ DEFAULT now()
);
```

You can then **index** these for fast traversal, and optionally mirror this into a Neo4j or graph store.

Define relation types you'll support initially

> "Edges" in the Zyme Graph must have **meanings** (types) so users and algorithms know what kind of connection they represent.

🧱 Start with a minimal but expressive set

Relation Type	Meaning	Example
`CHILD_OF`	Zyme A navigates *down into* Zyme B	"P vs NP" → "Cook–Levin Theorem"
`DEFINED_BY`	A keyword or term in Zyme A is explained in Zyme B (a glossary or explainer Zyme)	"NP-complete" → definition node

Relation Type	Meaning	Example
`RELATED_TO`	Zyme A is conceptually similar or topically linked to Zyme B	“Turing Machines” ↔ “Finite Automata”
`AUTHORED_BY`	The Zyme is authored by a specific person node	“Smith v. Jones” → “Justice Ginsburg”
`MENTIONED_IN`	Node B is mentioned or cited within Node A	“Statute 123” → “Opinion X”
`EVENT_AT`	The Zyme relates to an event, location, or date	“AAAI 2023” → “Conference on NLP Safety”

🔁 Directionality & symmetry

Relation	Direction	Inverse
`CHILD_OF`	A → B	`PARENT_OF`
`DEFINED_BY`	A → B	— (definition doesn't link back)
`RELATED_TO`	Symmetric	Self-inverse
`AUTHORED_BY`	A → B	`AUTHORED`
`MENTIONED_IN`	B → A	—

For v1, implement `CHILD_OF`, `DEFINED_BY`, and `RELATED_TO`.
Later you can expand to support `MENTIONED_IN`, `AUTHORED_BY`, etc.

🧠 How these types power UX

Feature	Depends on
“Backtrack to parent Zyme”	`CHILD_OF`
“View glossary definition”	`DEFINED_BY`
“Explore similar concepts”	`RELATED_TO`
“Show all Zymes by author X”	`AUTHORED_BY`
“Where is this concept cited?”	`MENTIONED_IN`

Wire UI hooks for graph traversal & hop-distance display

> Let users move across the knowledge graph—**not just within a single Zyme.**

1. Where do traversals start?

Any underlined word, icon, author tag, citation, or footnote can trigger a traversal:

- Click **keyword** → "Go to Zyme" (navigates to `DEFINED_BY`)
- Click **mini-menu** → "See related Zymes" (navigates `RELATED_TO`)
- Click **author name** → authored Zymes
- In **Compare mode**, pull nodes that share a `RELATED_TO` link

2. What are the traversal paths?

Graph traversal shows:

- **Child navigation** → user clicks “Go deeper” → `CHILD_OF`
- **Side navigation** → “Explore related” → `RELATED_TO`
- **Glossary navigation** → “Define term” → `DEFINED_BY`
- **User journey** → breadcrumb trail (from one Zyme to another)

Each path is based on a known relation type from the `zyme_edge` table.

3. Hop-distance (optional, future)

> This feature calculates how far one concept is from another (in number of graph edges).

UX Examples	Backend requirements
“This concept is 2 hops away from Cook–Levin Theorem.”	BFS or Dijkstra’s across the edge table
“How many hops connect ‘SAT’ to ‘Quantum Proofs’?”	`WITH RECURSIVE` SQL or native GraphDB

📌 Where to show it (future idea only)

- In a sidebar (“Concept proximity”)

- In a hover tooltip for a distant link
- In an “advanced” or “exploration mode” toggle

Not urgent, but should be built with an **edge-weight-aware graph structure** in mind.

4. UI components needed

Component	Function
Breadcrumb B (graph trail)	Show where user has traveled across nodes
“My Zyme Map”	Render graph of visited nodes
Mini-menu options	“Go to Zyme”, “Explore related”, “Define term”
Keyword preview	Hover on term → preview of target Zyme

C. Example Mapping

From JSON:

```
{
  "id": "zyme://doe2025-pnas",
  "levels": {
    "L0": [{ "id": "h1", "text": "Cook–Levin proves SAT ∈ NP.", "keywords": ["SAT", "NP"] }]
  },
  "edges": [
    { "target": "zyme://concept-SAT", "relation": "DEFINED_BY", "via": { "level": "L0", "keyword": "SAT"
} }
  ]
}
```

To DB:

- Insert ZymeNode into `zyme` table.
- Insert L0 card `h1` into `zyme_section` (with `section_type = 'overview'`)
- Insert each keyword as a `glossary_term` or `article_keyword`
- Insert edge:

```
INSERT INTO zyme_edge (
  from_id, to_id, relation, via_level, keyword
) VALUES (
  'zyme://doe2025-pnas',
  'zyme://concept-SAT',
  'DEFINED_BY',
  'L0',
  'SAT'
);
```

D. Why This Matters

- **You control how Zymes persist between sessions.**
- Enables **deep-linking**, **search**, **global graph traversal**, and **cross-doc relationships**.
- You unlock scalable UI features like "show all Zymes that reference NP-completeness" or “build a graph from my last 10 Zyme hops.”

Let me know if you want help:

- Designing the schema extension
- Choosing between SQL / Mongo / Neo4j
- Writing queries for common traversal tasks

This step makes Zymes live *as data*, not just static output.

B. Define relation types you’ll support initially (e.g. CHILD_OF, RELATED_TO).

C. Wire UI hooks for graph traversals & hop-distance displays.