**Hierarchical Graph Architecture**

* **Purpose**: Represents the hierarchical structure of the data, ensuring all entities are interconnected through parent-child relationships. I feel this is best for large data sets, which is what we will be connecting. Nodes and edges in neo4j graph builder:

[**https://data-importer.neo4j.io/?\_gl=1\*w4h04p\*\_gcl\_au\*MTcyODIxMjQ5Ny4xNzMyODczMTk4\*\_ga\*MjExNjcwODkwNi4xNzMyODczMTk4\*\_ga\_DL38Q8KGQC\*MTczMzIwMzE5My4zLjEuMTczMzIwMzI2Ny4wLjAuMA..\*\_ga\_DZP8Z65KK4\*MTczMzIwMzE5My4zLjEuMTczMzIwMzI2Ny4wLjAuMA**](https://data-importer.neo4j.io/?_gl=1*w4h04p*_gcl_au*MTcyODIxMjQ5Ny4xNzMyODczMTk4*_ga*MjExNjcwODkwNi4xNzMyODczMTk4*_ga_DL38Q8KGQC*MTczMzIwMzE5My4zLjEuMTczMzIwMzI2Ny4wLjAuMA..*_ga_DZP8Z65KK4*MTczMzIwMzE5My4zLjEuMTczMzIwMzI2Ny4wLjAuMA)**..**

* **Advantages**:
  + Provides a clear parent-child relationship.
  + Useful for navigating from a high-level overview (e.g., domain) to granular details (e.g., specific user stories or bugs).

Note: I think the following graphs can be used case by case or be implemented somewhere down the parent tree.

### **Feature-Centric Graph Architecture**

* **Purpose**: Focuses on connecting features, their qualities, and associated user stories, irrespective of hierarchical relationships.
* **Nodes**:
  + Feature
  + Quality
  + User Story
  + Acceptance Criteria
  + Common Bug
* **Edges**:
  + Feature → Quality
  + Quality → User Story
  + User Story → Acceptance Criteria
  + User Story → Common Bug
* **Advantages**:
  + Simplifies queries related to features and their impacts.
  + Enhances retrieval efficiency for scenarios focused on feature details.

### **User Story-Centric Graph Architecture**

* **Purpose**: Optimized for exploring user stories and their relationships with features, bugs, and acceptance criteria.
* **Nodes**:
  + User Story
  + Feature
  + Platform
  + Common Bug
  + Acceptance Criteria
* **Edges**:
  + User Story → Feature
  + User Story → Platform
  + User Story → Common Bug
  + User Story → Acceptance Criteria
* **Advantages**:
  + Highlights user needs and significant points.
  + Suitable for generating responses that prioritize user-specific insights.

**MongoDB Schema**

**Domain Collection**

"\_id": ObjectId("..."),

"name": "banking", // Domain name

"subdomains": [

"\_id": ObjectId("..."), // Subdomain reference

"name": "Retail",

"appxs": [

ObjectId("...") // Reference to Application (SBI\_YOLO)

**Applications Collection**

"\_id": ObjectId("..."),

"app\_name": "SBI\_YOLO",

"region": "India",

"platforms": [

ObjectId("...") // Reference to Platform (e.g., "mobile\_Andr")

**Platforms Collection**

"\_id": ObjectId("..."),

"platform": "mobile\_Andr", // Platform name

"software\_type": "Frontend", // Frontend or Backend

"features": [

ObjectId("..."), // Reference to Feature (e.g., "authentication")

**Features Collection**

"\_id": ObjectId("..."),

"name": "authentication", // Feature name

"type": "functional", // Type: "functional" or "non-functional"

“Standard”: “Good” // standard “good” or bad“”

"quality": [

ObjectId("...") // Reference to User Story

**User Stories Collection**

"\_id": ObjectId("..."),

"value": "As a retail customer in India, I want to securely authenticate myself using my mobile device on the SBI\_YOLO app, So that I can access my bank account and perform transactions safely.",

"acceptance\_criteria": [

"AC1: Authentication screen loads in less than 2 seconds.",

"AC2: User can successfully login with username and password."

],

"bugs": [

ObjectId("...") // Reference to Bug

**Bugs Collection**

"\_id": ObjectId("..."),

"description": "No clear error message for invalid username or password.", // Bug description

"type": "authentication" // Type of bug (e.g., "authentication", "UI", etc.)

**Acceptance Collection (optional can stay in User Stories Collection)**

"\_id": ObjectId("..."),

"criteria": "AC1: Authentication screen loads in less than 2 seconds."