✅ Lab: Document a GenAI-RAG Architecture and Manage It in JIRA

🧩 Part 1: Visualize the GenAI-RAG Architecture Using Mermaid or Draw.io

---

🔹 Step 1: Define the Use Case

Choose a GenAI use case, for example:
"AI-powered customer support assistant with Retrieval-Augmented Generation (RAG)."

This system will:

- Accept natural language questions from users
- Retrieve relevant documents
- Generate helpful responses using a Large Language Model (LLM)

🔹 Step 2: Identify System Components

List the major components:

- UI Layer: Chat interface
- LLM Layer: OpenAI GPT, Mistral, Claude, etc.
- Vector Store: FAISS, Pinecone (for document embeddings)
- Knowledge Graph: RDF store like Stardog or Neo4j with SPARQL access
- RAG Pipeline: Combines vector + graph search
- API Gateway: Routes requests
- Agent Orchestrator (Optional): LangGraph, AutoGen, or CrewAI
- Monitoring: Prometheus / CloudWatch
- Deployment: AWS Fargate / Azure Container Apps

🔹 Step 3: Draw the Architecture

✅ Option A: Use Draw.io

1. Open draw.io
2. Choose **Blank Diagram**
3. Add boxes for each component
   - Use arrows to represent flow:

        - User input → API Gateway → RAG Engine → Vector DB / Graph DB → LLM → Output

4. Group components logically:
    - Retrieval Layer (Vector + Graph)
    - Inference Layer (LLM)
    - Orchestration Layer (Optional Agents)

5. Label each component clearly

6. Export as PNG, PDF, or share via link

✅ Option B: Use Mermaid (in Markdown or VS Code)

```
graph TD
    UI[Chat UI] --> API[API Gateway]
    API --> RAG[RAG Engine]
    RAG --> Vector[Vector DB (Pinecone)]
    RAG --> Graph[Knowledge Graph (SPARQL)]
    RAG --> LLM[LLM (OpenAI)]
    LLM --> Output[Response Generator]
    Output --> UI
```

> You can paste this into any Mermaid live editor like [Mermaid Live Editor]() or VS Code with the Mermaid extension.

🛠️ Part 2: Create a Sample Sprint Board and Roadmap in JIRA

🔷 Step 1: Create a JIRA Project

1. Go to [https://jira.atlassian.com]() or your team's JIRA instance

2. Click **Create Project**

3. Select **Scrum** or **Kanban** template (Scrum preferred for sprints)

4. Name your project: *"GenAI RAG Support Assistant"*

🔷 Step 2: Define Epics (High-Level Goals)

Create Epics to match system modules:

- Epic 1: Build RAG Pipeline
- Epic 2: Integrate Knowledge Graph
- Epic 3: LLM Prompt Engineering & Testing
- Epic 4: Deploy on Cloud (AWS Fargate)
- Epic 5: Monitoring & Evaluation

🔹 Step 3: Add Stories and Tasks

Example tasks under **Epic 1: Build RAG Pipeline**:

- Create document embedding pipeline for Pinecone
- Load documents and run similarity search
- Connect GraphDB and build SPARQL template
- Combine top-k from vector and graph into context window
- Route final prompt to OpenAI and log output

Include:

- Task descriptions
- Acceptance criteria (e.g., “returns relevant response in <2 sec”)
- Labels: `vector`, `graph`, `LLM`, `API`

🔹 Step 4: Configure Sprint Board

1. Go to your **Board Settings**
2. Click **Create Sprint**
3. Drag stories into the sprint
4. Set a 2-week timeframe
5. Add **Story Points** to estimate effort
6. Assign tasks to team members

🔹 Step 5: Add Milestones to the Roadmap

Under **Roadmap**:

- Week 1: Vector DB up and running with embeddings
- Week 2: Graph querying integrated with SPARQL
- Week 3: LLM prompt refinement complete
- Week 4: Pipeline tested end-to-end
- Week 5: Cloud deployment and logging active

📦 Deliverables

- ✅ **Architecture diagram** (.drawio, .png, or Mermaid markdown)
- ✅ **JIRA board** with epics, tasks, and roadmap

- ✅ **Sprint setup** with estimation and ownership
- ✅ Optional: Export sprint report or burndown chart
