uminai Technical Knowledge

Introductory Demo Repository:

uminai MVP Frontend: uminai-frontend-app (click to open)
Requirements:

- Nodejs v20+ and above

uminai Backend Service Provider: uminai-backend-service-provider (click to open)
Requirements:

- Nodejs v20+ and above
- Mongodb instance (You may setup local, Atlas or Decentralised)

Running "**uminai MVP Frontend**"

1. npm install --legacy-peer-deps
2. update the backend url in "**src\shared\consts.ts**"
   a. BASE_SERVER_URL={{**TO_YOUR_BACKEND_ENDPOINT**}}
3. npm run dev to start the application

Running "**uminai Backend Service Provider**"

1. npm install
2. Create .env file (Refer from .env.example)
   a. OPENAI_API_KEY
   b. OPENAI_API_URL
   c. CHAINSAFE_KEY
   d. CHAINSAFE_API_KEY
   e. CHAINSAFE_BUCKET_ID
   f. MONGO_URI
3. npm run server

Introductory Demo Repository:

This demo covers:

1. uminai Public Library of Product Information
2. uminai Decentralised Identity (uDID) for Physical Entity
3. uminai Decentralised Product Data (uDPD)

Objective in this document is to demonstrate on of the proof of concept for uminai core implementations.

In this document, we will guide you through how we the setup from is done for participating in Paris Blockchain Week Hackathon 2025, in collaboration by XRPL Commons. Therefore, our implementation focused on XRPL network.

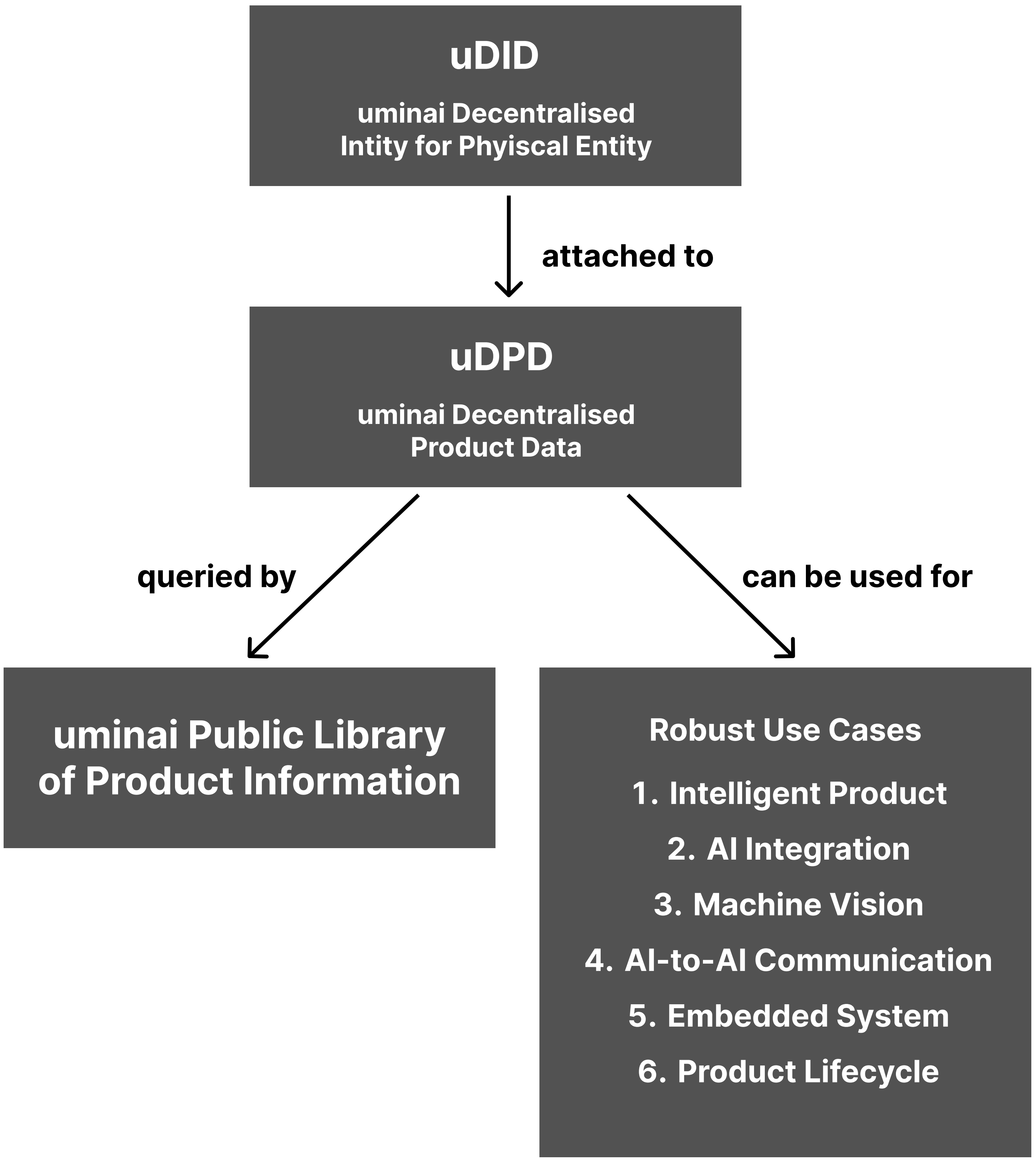
Things to acknowledge:

1. uminai leverages XRPL EVM (Sidechain) - Fast and secure independent ledger with EVM compatibility
2. Due to multiple domains and resources for EVM-based ecosystem and libraries supported, it is our common goal and perspective that XRPL EVM-based would be able to achieve our goals.
3. Last not least, EVM-based allows cross-chain application for scaling opportunity.
4. This is only MVP Products. Therefore, we ensure continuous update and enhancement to achieve our goals.

Continue next page.

Introductory Demo Repository:

The flow of uminai ecosystem



Introductory Demo Repository:

Public Library of Product Information

Key feature:

1. A open public knowledge of product information that is abstracted from json schema, combining both unstructured and structured.
2. Ensuring high-granularity, transparency and availability in decentralised storage

Use cases:

1. Intelligent Integration: uminai Public Library to serve AI, Robotics and machine-readable system to interact with physical entities and understanding the context.
2. Enhanced Product Discovery: Supports dynamic product searches by leveraging detailed attributes and metadata.
3. Data Analytics & Insights: Integrates with user interaction systems to analyze product performance and consumer engagement.

Technologies Used:

- Off-chain Backend (uminai Service Provider): Node.js
- API Development: RESTful
- Data Processing: Mongodb with Atlas Vector Search
- LLM: Mistral/Qwen2.5-7b for Query Processing

---

uminai Decentralised Identity (uDID) for Physical Entity

Key feature:

1. Unique Digital Identity: Assigns a unique identifier to each physical entity, ensuring global uniqueness and traceability.
2. Interoperability: Designed to work with existing IoT, AI, and industrial systems, enabling easy integration.

Key feature:

1. Asset Tracking & Management: Enhances traceability of products across the supply chain by linking physical objects to their digital identities.
2. Smart Manufacturing & IoT: Allows automated systems to interact with physical entities based on their identity attributes, driving efficiency in industrial settings

Introductory Demo Repository:

uminai Decentralised Identity (uDID) for Physical Entity (Continue)

Technologies Used:

- Decentralised Storage & Ledger: XRPL EVM for uDID Management
  - Create | Update | Revoke
- IPFS via Chainsafe: Pinning CID hash for uDID Document Lookup and Resolver

---

uminai Decentralised Product Data

Key feature:

1. Unified Data Model:
   a. Abstracts product information from various JSON schemas into a standardized, unified format.
   b. Supports integration of structured attributes (e.g., SKU, dimensions, pricing) and unstructured details (e.g., descriptions, reviews).
2. Interoperability: Using NoSQL (MongoDB) structure to ensure machine-readable format and query with Vector Search or RAG supported mechanism.

Use cases:

1. Supply Chain and Inventory Management
2. E-commerce and Retail
3. Cross integration for Intelligent Product, Embedded System that supports JSON Schema for better readability.

Technologies Used:

- Current Stage: MongoDB as a NoSQL database to manage diverse product data formats efficiently.
- Future Vision: Decentralized storage solutions inspired by blockchain or distributed ledger technologies to decentralize MongoDB’s functionalities.

End of Document (5-MARCH-2025)