Paradigm App – Steps Needs to follow to build the required services

Company: Element Zero Labs

Author : Amit Yadav (Automation Engineer)

Watch these videos for better understanding :

Explained reference documents

Explained steps you have to follow

🔍 REFERENCE DOCUMENTATION & RESOURCES

Before starting development, refer to these documents and schemas:

1. For Backend Development Reference:

- **Black Services & Green Services API endpoints :** Refer to "Paradigm Build docs" in Google Drive
- **For MCP Building:** Refer to Paradigm MCP Build docs in Google Drive .

2. Required System Component Development & Data Flow Schema Refer (Miro):

- **Black Services:**
    - User ID generation system component schema

    - Developer Group Workspace ID generation system component schema
- **Green Services Groups workspace Chats:**
    - Developer Groups / Group 1, 2, 3 / Chat(s) . All ID generation system component schema **(Refer to Miro Board)**
- **Green Services Groups workspace , Projects /Chats :**
    - Developer Groups / Group 1, 2, 3 / Project(s) / Chat(s) / KWB(s) + INN . All ID generation system component schema **(Refer to Miro Board)**

3. Overall Architecture Reference:

- **PARADIGM APP ARCHITECTURE** in Miro - Review complete build and test architecture

4. Important implementation rules : Refer this docs

👉**Important 1 :** You need to connect provided cloud database with your backend services , so that we can do the testing on cloud directly.

Database Link :

👉**Important 2 :** After building your All your backend services & Webhooks provide your backend boiler plate in below github repository to deploy them on cloud environment for Real life testing

Github Repo link :

👉**Important 2 :** After building your webhooks connected with your endpoints provide webhooks to Amit for integrating in frontend and n8n

🚨 **CRITICAL Instruction :**

Backend Self-Trigger Implementation

You have to implement a function that detects when your backend is running in a container environment and automatically triggers initialization immediately after server startup.

👉FOLLOW THESE STEPS TO BUILD PARADIGM API & Services

All development will be done in next.js

Step 1: Build Black Services API endpoint for User Authentication

Build a comprehensive authentication system that handles all user management and access control for the Paradigm App.

→ After building all black services Api, build a dedicated custom webhook that connects to all Black services api endpoints

→ Test Black Services webhook in your development environment working with all the custom webhook.

Step 2: Build Green Services APIs and Build Green Services Webhook

→ After building all Green Services APIs, build Green Services webhook that connects all green services endpoints into single custom webhook

→ Test Green Services API endpoint with the dedicated custom webhook in your development environment to ensure all The connection between the webhooks are working properly .

- **Step 3: Provide dependency logic between all the backend services :**

  1- Black services end point API :

  > With Green services API :

  - Black Services provides user authentication to Green Services
  - Black Services generates system unique IDs that Green Services uses
  - Black Services validates user sessions for Green Services operations
  - Black Services supplies user_id and group_id to Green Services
  - Black Services handles role-based permissions for Green Services access

  > N8N :

  - Black Services sends auth responses through N8N back to frontend
  - Black Services relies on N8N to route requests to correct endpoints
  - Black Services depends on N8N for API security and validation

2- Green services end point API :

> With Black services : (Refer to the above)

> With MCP:

- Green Services routes project operations to MCP for context aware output
- Green Services sends KWB content and instructions to MCP
- Green Services depends on MCP for context aware output
- Green Services requires MCP for creating chats under project with knowledge base and instruction context

> N8N :

- N8N forwards frontend content requests to Green Services
- N8N routes simple operations directly to Green Services
- N8N routes complex operations to Green Services → MCP flow

> LLM Anthropic : (done by Amit Yadav)

- Green Services sends chat messages to Anthropic API for AI responses
- Green Services depends on Anthropic API for Claude AI processing
- Green Services requires Anthropic API for both direct chats and MCP related chats
- Green Services formats user prompts for Anthropic API consumption

Step 4: Build MCP Enhancement to Green Services

Build MCP (Model Context Protocol) that makes **SPECIFIC** Green Services api to enrich chats messaging context with knowledge base Docs (PDF , WORDS ,) & instruction context.

→ After Building & completing the MCP in conjunction with green services specific end points and required data flow , provide & integrate dedicated custom MCP webhook in Next.js in VS Code

→ Test that MCP operations provide better AI responses using knowledge base information

When & where MCP (Model Context Protocol) would be Required:

Developer Groups / Group 1, 2, 3 / Project(s) / Chat(s) -KWB(s) + INN

- User creates projects within developer groups
- User adds knowledge bases (KWB) to projects
- User adds instructions (INN) to knowledge bases
- User create chats under projects that reference chats context to one of the list of knowledge bases docs that previously trained agents via MCP.

‘

Step 5: Implement required Security for All Backend Services

→ Provide required API key for integration between "n8n- & backend-api services "

→ Verify unauthorized user login are rejected and authorized users are able to access the app

Step 6: Cloud Databases integration : With All Backend end point services + MCP server

Required Databases which you have to integrate with your backend services for Cloud Testing Environment :

1- NoSql Database (Cloud Instance) : Link

2- Storage Database (Cloud Instance) : Link

→ Integrate provided cloud database links directly into backend services

→ Configure database connections for:

- Black Services (authentication data in NoSQL)
- Green Services (project, chat, and content data)
- MCP Server (knowledge base and instruction storage)

→ Implement proper connection pooling and error handling for cloud database connections

→We will Test all database operations directly on cloud infrastructure

→ Verify all Next.js webhook operations return proper data from live cloud databases for authentication (NoSQL) and content operations (Storage)

Step 7: Test Everything Together on cloud environment

Run complete testing of all backend services across :

Webhooks for :
>Black Service endpoints
>Green Services endpoints
>MCP server

Carry out Test & Verification on Cloud Environment as follow :

Developer → Test All backend services dataflow are working properly on cloud:

1- generate all system ID generation , and
2- the critical function between the above backend services
3-Chats are creating with proper user id and group id
4-Project can be created and knowledge base and instructions are uploading
5-Also files uploading under the chats have proper unique id etc.

"Developer "→ Test all developed webhooks are integrated and working with all the above services endpoints on cloud environment

"Amit Yadav "→ Provide the developed webhooks to AMIT YADAV . so, Amit shall integrate them in Fronted and N8N then the working pipeline between the Frontend , N8N & backend must be tested accordingly for these webhooks to communicate & complete handshake on live cloud environment

"Developer "→ Test the Pipeline between all the cloud-deployed backend services , N8N and LLM Anthropic where the backend services must successfully carry API calls and loopback feed from the LLM API.

Step 8: Testing with frontend

→ **After verifying the Step-7 (including cloud deployment verification),**

Backend developer must provide all developed webhooks to Amit Yadav for frontend integration

→ Amit Yadav will integrate the provided webhooks into frontend and deploy them on cloud

→ Once both frontend and backend are deployed on cloud infrastructure, meet with Amit Yadav for comprehensive cloud-based testing

→Amit shall link the cloud-deployed App FE to N8N, at this point we run a live test by testing the app product features working in real-time on cloud environment

If this cloud-based test across all required product features is successful, the candidate has passed the hands on technical test

---

🚨 **CRITICAL Instruction :**

Backend Self-Trigger Implementation

You have to implement a function that detects when your backend is running in a container environment and automatically triggers initialization immediately after server startup.