1. Requirements Gathering

Functional Requirements

- 1. **Plug-in Registration:** Ability to register new plug-ins with metadata (name, version, scanner model, etc.).
- 2. **Plug-in Storage:** Store plug-ins in cloud storage (AWS S3/Azure Blob/**GCS**) and index metadata in a database.
- 3. Plug-in Validation: Validate uploaded plug-ins before allowing execution.
- 4. **API for Plug-in Management:** gRPC-based APIs for interacting with the Plug-in Manager.
- 5. Lifecycle Management: Enable updates, versioning, and deprecation of plug-ins.
- 6. Microservice Interactions
 - a. Plugin Manager interacts with:
 - i. ADC Backend (via gRPC)
 - ii. Cloud Storage (AWS S3/ Azure Blob/ GCS)
 - iii. Metadata Database (MySQL)
 - b. Cross communication among other microservices
 - i. Configuration Manager Service and Device Manager Service

Non-Functional Requirements

- Scalability: The system should handle multiple scanner models and firmware versions.
- Performance: Retrieve plugin related request data ASAP
- **Security:** Secure API endpoints with authentication & authorization. (only Life cycle management)
- Admin UI for management
- **High Availability:** Deploy as a microservice with failover mechanisms.

2. High-Level Architecture (Design)

1. Overall Architecture

- Frontend: React-based UI (consuming JSON).
- HAL Backend: Communicates via gRPC with ADC Backend.
- ADC Backend: Manages business logic and delegates plugin management.
- Plug-in Manager: Manages plugin repository, validation, and metadata.
- Cloud Storage: Stores plugin files (images, metadata, etc.).

2. Plug-in Manager Features & gRPC APIs

Feature	gRPC API	REST Endpoint (for UI)
Import Plugin	ImportPlugin()	POST /import
Download Plugin	DownloadPlugin()	GET /download/{id}
Get all plugin families	GetAllPluginFamilies()	GET /plugin-families
Get plugins of family	GetPluginsOfFamily()	GET /plugin- families/{familyId}/plugins
Get plugin revisions of plugin	GetPluginRevisions()	GET /plugins/{pluginId}/revisions
Query by Model	GetPluginsByModel()	GET /query?model={model}
Query by Model & FW	GetPluginsByModelAndFirmwar e()	GET /query?model={model}&fw={fw }
Delete Plugin	DeletePlugin()	DELETE /plugin/{id}

3. Cloud Storage & Metadata Management

- Store plugin files in GCS.
- Maintain a **metadata index** in **MySql**:(Every plugin revision having Metadata.xml file)

```
<description>2D Imager</description>
<firmware>PAADFS00-007-R00</firmware>
<firmware-naming-style>2</firmware-naming-style>
<revision>038</revision>
<release-type>STANDARD</release-type>
<release-date>1/26/2024</release-date>
<build>2</build>
<plugin-schema-version>1.0000</plugin-schema-version>
<rsm-version>2.0</rsm-version>
<models>
 <model>CR2278-PC10004CN</model>
  <model>CR2278-PC10004WW</model>
  <model>DS2278-SR00006ZZWW</model>
  <model>DS2278-SR00007ZZCN</model>
  <model>DS2278-SR00007ZZWW</model>
  <model>DS2278-SR00007ZZY</model>
  <model>DS2278-SR00267ZZWW</model>
  <model>DS2278-SR7U2100PRC</model>
  <model>META-BKP</model>
  <model>META-BKW</model>
</models>
<combined-firmware name="SAADFS00-007-R00">
 <component name="Decoder">PAADFS00-007-R00</component>
 <component name="DS2278 Top Level">CAADFS00-007-R00</component>
 <component name="Cradle">PAADNS00-006-R01</component>
 <component name="CR2278 Top Level">CAADNS00-006-R01/component>
</combined-firmware>
```

As example in .config file having

```
<plug-in build="2">DS2278-STANDARD SR MODELS + CRADLE-S-038</plug-in>
```

This variable is built by combining family, name and revision of metadata.

4. gRPC API Design for Plug-in Manager

gRPC Service Definition

```
syntax = "proto3";
service PluginManager {
  rpc ImportPlugin(PluginRequest) returns (PluginResponse);
```

```
rpc DownloadPlugin(PluginDownloadRequest) returns (PluginFile);
rpc GetAllPluginFamilies(PluginFamiliesRequest) returns (PluginFamiliesResponse);
rpc GetPluginsOfFamily(GetPluginsOfFamilyRequest) returns
(GetPluginsOfFamilyResponse);
rpc GetPluginRevisions(GetPluginRevisionsRequest) returns
(GetPluginRevisionsResponse);
rpc GetPluginsByModel(PluginQueryModelRequest) returns (PluginListResponse);
rpc GetPluginsByModelAndFirmware(PluginQueryFirmwareRequest) returns
(PluginListResponse);
rpc DeletePlugin(PluginDeleteRequest) returns (PluginResponse);
```

3. Microservices & Plug-in Interaction

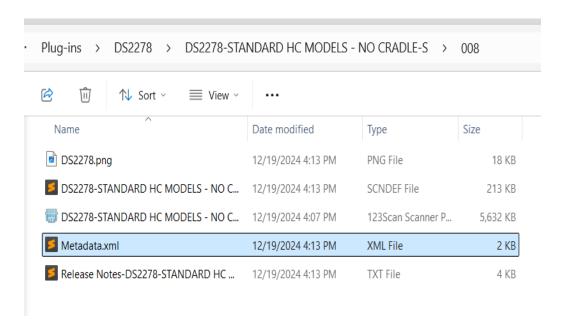
The **Plug-in Manager** will expose gRPC APIs and interact with **cloud storage and metadata DB**.

Workflow

- 1. Register Plugin
 - a. A new **plugin file** is uploaded.
 - b. The metadata is extracted and stored in PostgreSQL/MySQL.
 - c. The plugin is stored in GCS.
- 2. Validate Plugin (internal)
 - a. Checks schema compatibility.
 - Ensures firmware and model match.
- 3. List Plugins
 - a. Queries the **database** for available plugins.
 - b. Returns key: value pairs with plugin details.

Eg: image: imageUrl

scndefFile: scndefFileUrl ... like wise for all followings



4. Download Plugin Fetches the plugin from cloud storage.

UML - Class Diagram

UML - Sequence Diagram