HLD Appendix – OpenAPI Skeletons & Data Model

This appendix includes OpenAPI YAML skeletons for key services, the Mermaid class diagram for the data model, and an SVG rendering for convenience.

# OpenAPI Skeletons

## identity-svc

openapi: 3.1.0  
info:  
 title: KisanConnect Identity Service  
 version: 0.1.0  
servers:  
 - url: https://api.kisanconnect.example/v1  
paths:  
 /auth/otp/send:  
 post:  
 summary: Send login OTP  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 phone: { type: string, example: '+91XXXXXXXXXX' }  
 responses:  
 '202':  
 description: OTP dispatched  
 /auth/otp/verify:  
 post:  
 summary: Verify OTP and issue tokens  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 phone: { type: string }  
 code: { type: string }  
 responses:  
 '200':  
 description: Auth tokens  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/AuthTokens'  
components:  
 schemas:  
 AuthTokens:  
 type: object  
 properties:  
 access\_token: { type: string }  
 refresh\_token: { type: string }  
 expires\_in: { type: integer }

## profile-svc

openapi: 3.1.0  
info:  
 title: KisanConnect Profile Service  
 version: 0.1.0  
paths:  
 /farmers:  
 post:  
 summary: Create/update farmer profile  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema: { $ref: '#/components/schemas/Farmer' }  
 responses:  
 '200': { description: Upserted }  
 get:  
 summary: List farmers (admin)  
 parameters:  
 - in: query  
 name: org\_id  
 schema: { type: string }  
 responses:  
 '200':  
 description: OK  
 /farmers/{id}:  
 get:  
 summary: Get farmer by id  
 parameters:  
 - in: path  
 name: id  
 required: true  
 schema: { type: string }  
 responses:  
 '200':  
 description: OK  
 content:  
 application/json:  
 schema: { $ref: '#/components/schemas/Farmer' }  
components:  
 schemas:  
 Farmer:  
 type: object  
 properties:  
 id: { type: string }  
 phone: { type: string }  
 locale: { type: string }  
 pincode: { type: string }  
 org\_id: { type: string }  
 land\_size: { type: number }  
 lat: { type: number }  
 lon: { type: number }

## weather-svc

openapi: 3.1.0  
info:  
 title: KisanConnect Weather Service  
 version: 0.1.0  
paths:  
 /weather/forecast:  
 get:  
 summary: 7-day hyperlocal forecast  
 parameters:  
 - in: query  
 name: lat  
 required: true  
 schema: { type: number }  
 - in: query  
 name: lon  
 required: true  
 schema: { type: number }  
 responses:  
 '200':  
 description: Weather forecast

## prices-svc

openapi: 3.1.0  
info:  
 title: KisanConnect Prices Service  
 version: 0.1.0  
paths:  
 /prices:  
 get:  
 summary: Get mandi prices for crop near location  
 parameters:  
 - in: query  
 name: crop  
 required: true  
 schema: { type: string }  
 - in: query  
 name: lat  
 schema: { type: number }  
 - in: query  
 name: lon  
 schema: { type: number }  
 responses:  
 '200': { description: OK }  
 /alerts/prices:  
 post:  
 summary: Create price alert  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 crop: { type: string }  
 min\_rate: { type: number }  
 radius\_km: { type: number }  
 responses:  
 '201': { description: Created }

## advisory-svc

openapi: 3.1.0  
info:  
 title: KisanConnect Advisory Service  
 version: 0.1.0  
paths:  
 /advisories:  
 post:  
 summary: Submit crop image for advisory  
 requestBody:  
 required: true  
 content:  
 multipart/form-data:  
 schema:  
 type: object  
 properties:  
 image: { type: string, format: binary }  
 plot\_id: { type: string }  
 responses:  
 '200':  
 description: Advisory response  
 content:  
 application/json:  
 schema: { $ref: '#/components/schemas/Advisory' }  
 /advisories/{id}:  
 get:  
 summary: Fetch advisory  
 parameters:  
 - in: path  
 name: id  
 required: true  
 schema: { type: string }  
 responses:  
 '200': { description: OK }  
components:  
 schemas:  
 Advisory:  
 type: object  
 properties:  
 id: { type: string }  
 image\_ref: { type: string }  
 inferences:  
 type: array  
 items:  
 type: object  
 properties:  
 label: { type: string }  
 confidence: { type: number }  
 remedies:  
 type: object  
 properties:  
 organic: { type: string }  
 chemical: { type: string }  
 payload\_sha256: { type: string }  
 fabric\_tx: { type: string }  
 anchor\_tx: { type: string }

## schemes-svc

openapi: 3.1.0  
info:  
 title: KisanConnect Schemes Service  
 version: 0.1.0  
paths:  
 /schemes:  
 get:  
 summary: Search government schemes  
 parameters:  
 - in: query  
 name: state  
 schema: { type: string }  
 - in: query  
 name: category  
 schema: { type: string }  
 - in: query  
 name: q  
 schema: { type: string }  
 responses:  
 '200': { description: OK }

## forum-svc

openapi: 3.1.0  
info:  
 title: KisanConnect Forum Service  
 version: 0.1.0  
paths:  
 /threads:  
 post:  
 summary: Create thread  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 title: { type: string }  
 tags: { type: array, items: { type: string } }  
 responses:  
 '201': { description: Created }  
 get:  
 summary: List threads  
 responses:  
 '200': { description: OK }  
 /posts:  
 post:  
 summary: Create post in a thread  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 thread\_id: { type: string }  
 body: { type: string }  
 responses:  
 '201': { description: Created }

## ledger-svc

openapi: 3.1.0  
info:  
 title: KisanConnect Ledger Service  
 version: 0.1.0  
paths:  
 /ledger:  
 post:  
 summary: Add ledger entry  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 category: { type: string }  
 amount: { type: number }  
 date: { type: string, format: date }  
 responses:  
 '201': { description: Created }  
 get:  
 summary: List ledger entries  
 responses:  
 '200': { description: OK }

## iot-svc

openapi: 3.1.0  
info:  
 title: KisanConnect IoT Service  
 version: 0.1.0  
paths:  
 /iot/devices/register:  
 post:  
 summary: Register IoT device  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 device\_id: { type: string }  
 type: { type: string }  
 responses:  
 '201': { description: Created }  
 /iot/readings:  
 get:  
 summary: Fetch readings by device/time  
 parameters:  
 - in: query  
 name: device\_id  
 schema: { type: string }  
 - in: query  
 name: from  
 schema: { type: string, format: date-time }  
 responses:  
 '200': { description: OK }

## notary-svc

openapi: 3.1.0  
info:  
 title: KisanConnect Notary Service  
 version: 0.1.0  
paths:  
 /notary/hash:  
 post:  
 summary: Notarize payload hash on Fabric  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 ownerId: { type: string }  
 type: { type: string }  
 sha256: { type: string }  
 responses:  
 '200':  
 description: Fabric tx  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 fabric\_tx: { type: string }  
 /anchor/run:  
 post:  
 summary: Trigger batch anchor to public chain  
 responses:  
 '202': { description: Anchor scheduled }

## notification-svc

openapi: 3.1.0  
info:  
 title: KisanConnect Notification Service  
 version: 0.1.0  
paths:  
 /notify:  
 post:  
 summary: Send notification (SMS/Push)  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 channel: { type: string, enum: [SMS, PUSH] }  
 to: { type: string }  
 template\_id: { type: string }  
 params: { type: object }  
 responses:  
 '202': { description: Queued }

## media-svc

openapi: 3.1.0  
info:  
 title: KisanConnect Media Service  
 version: 0.1.0  
paths:  
 /media/sign:  
 post:  
 summary: Create pre-signed URL for upload  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 contentType: { type: string }  
 responses:  
 '200':  
 description: Signed URL  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 url: { type: string }  
 expiresIn: { type: integer }

# Data Model – Mermaid Class Diagram (source)

## DataModel\_ClassDiagram.mmd

classDiagram  
 class farmer {  
 +string id  
 +string phone  
 +string locale  
 +string pincode  
 +string org\_id  
 +float land\_size  
 +float lat  
 +float lon  
 }  
 class plot {  
 +string id  
 +string farmer\_id  
 +string soil\_type  
 +float area  
 +string crop  
 +string irrigation  
 +float lat  
 +float lon  
 +string season  
 +string variety  
 }  
 class advisory {  
 +string id  
 +string farmer\_id  
 +string plot\_id  
 +string image\_ref  
 +json inferences  
 +json remedies  
 +string model\_version  
 +string payload\_sha256  
 +string fabric\_tx  
 +string anchor\_tx  
 }  
 class ledger\_entry {  
 +string id  
 +string farmer\_id  
 +string plot\_id  
 +string category  
 +number amount  
 +date date  
 +string notes  
 }  
 class harvest {  
 +string id  
 +string farmer\_id  
 +string plot\_id  
 +string crop  
 +string variety  
 +date date  
 +number qty  
 +string unit  
 +number price  
 +string buyer  
 }  
 class market\_price {  
 +string id  
 +string mandi\_id  
 +string crop  
 +date date  
 +number modal  
 +number min  
 +number max  
 +string source  
 }  
 class iot\_reading {  
 +string device\_id  
 +datetime ts  
 +number soil\_moisture  
 +number temp\_c  
 +number vbat  
 +string plot\_id  
 }  
 class scheme {  
 +string id  
 +string state  
 +string category  
 +string title  
 +string description  
 }  
 class forum\_thread {  
 +string id  
 +string farmer\_id  
 +string title  
 +string[] tags  
 }  
 class forum\_post {  
 +string id  
 +string thread\_id  
 +string author\_id  
 +string body  
 +string status  
 +number toxicity\_score  
 +string lang  
 }  
  
 farmer "1" --> "many" plot : owns  
 farmer "1" --> "many" advisory : requests  
 plot "1" --> "many" advisory : related  
 farmer "1" --> "many" ledger\_entry : records  
 farmer "1" --> "many" harvest : produces  
 plot "1" --> "many" harvest : from  
 forum\_thread "1" --> "many" forum\_post : contains

# Data Model – SVG Rendering

[Open SVG directly: /mnt/data/DataModel\_ClassDiagram.svg] ()