African Bamboo - ODK External Validations

The Akvo External ODK validation application is an Android application that integrates with ODK Collect (Kobo Collect) to provide advanced validation capabilities for parcel boundary mapping in Ethiopia. The application leverages ODK's external app integration mechanism to perform server-side validations and return dynamic error messages to the Kobo Collect user.

- Project Sheet
- Low Level Design

Project Sheet

Name	African Bamboo - ODK External Validation
Project Scope	The African Bamboo - ODK External validation application is an Android application that integrates with ODK Collect (Kobo Collect) to provide advanced validation capabilities for parcel boundary mapping in Ethiopia. The application leverages ODK's external app integration mechanism to perform server-side validations and return dynamic error messages to the Kobo Collect user.
Contract Link	Link to the signed PDF document (PandaDoc)
PMT Link	Link to the Project Dashboard file in Google Drive
Start Date	Start date according to the contract
End Date	End date according to the contract
Repository Link	https://github.com/akvo/african-bamboo-odk-external-validations/
Tech Stack	List of technologies used to execute the technical scope of the project: • Front • Back • BD • Testing • CI • Hosting
Asana Link	2563 - African Bamboo Pilot Engineering Planning Board
Slack Channel Link	Link to the main Slack channel (proj-*-general)

Low Level Design

Validation Development

Basic Plot Validations

XLSForm

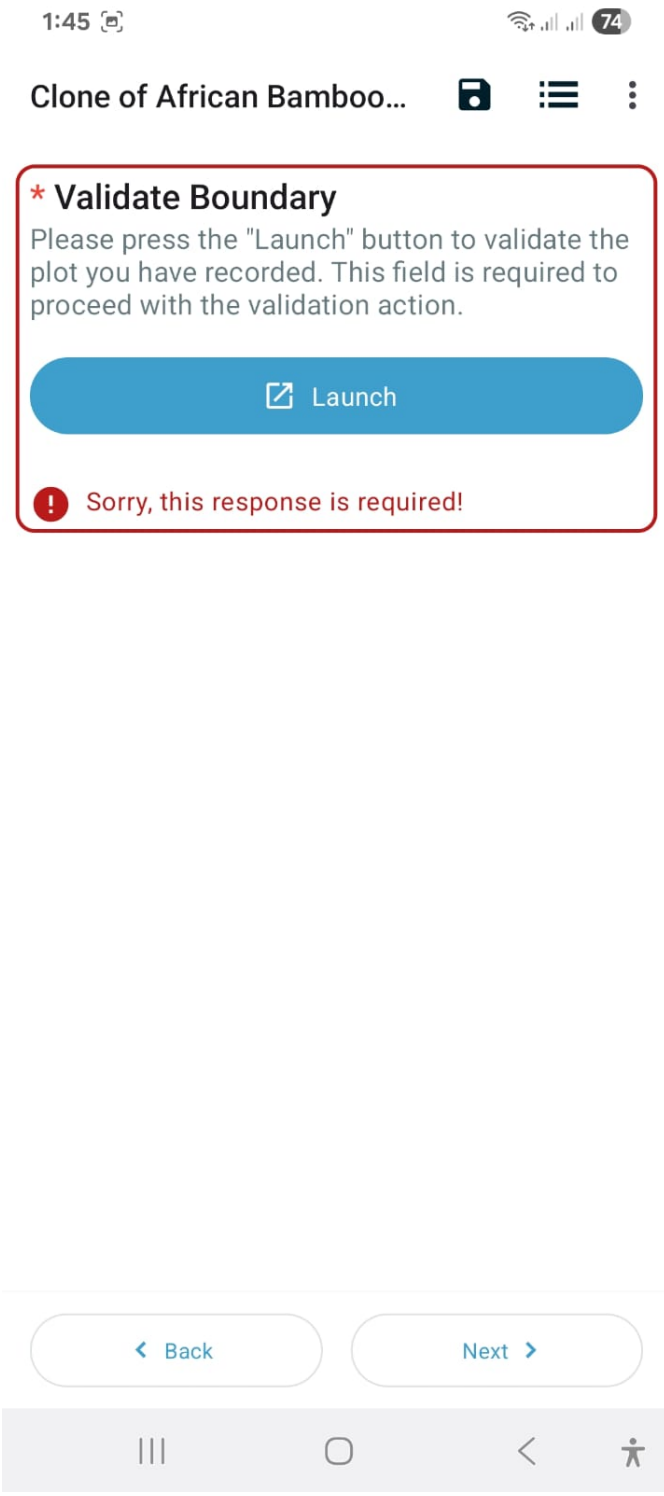
Field	Value	Description
type	text	
name	validate_polygon	
required	true	
appearance	ex:org.akvo.afribamodkvalidator.VALIDATE_POLYGON(shape=${target_field})	
constraint	. = ${manual_boundary}	
constraint_message	Please revalidate by clicking "Launch" button	

Full source: Spreadsheet - African Bamboo B.V. Parcel Boundary Mapping

Geometry Validation Logic and Response

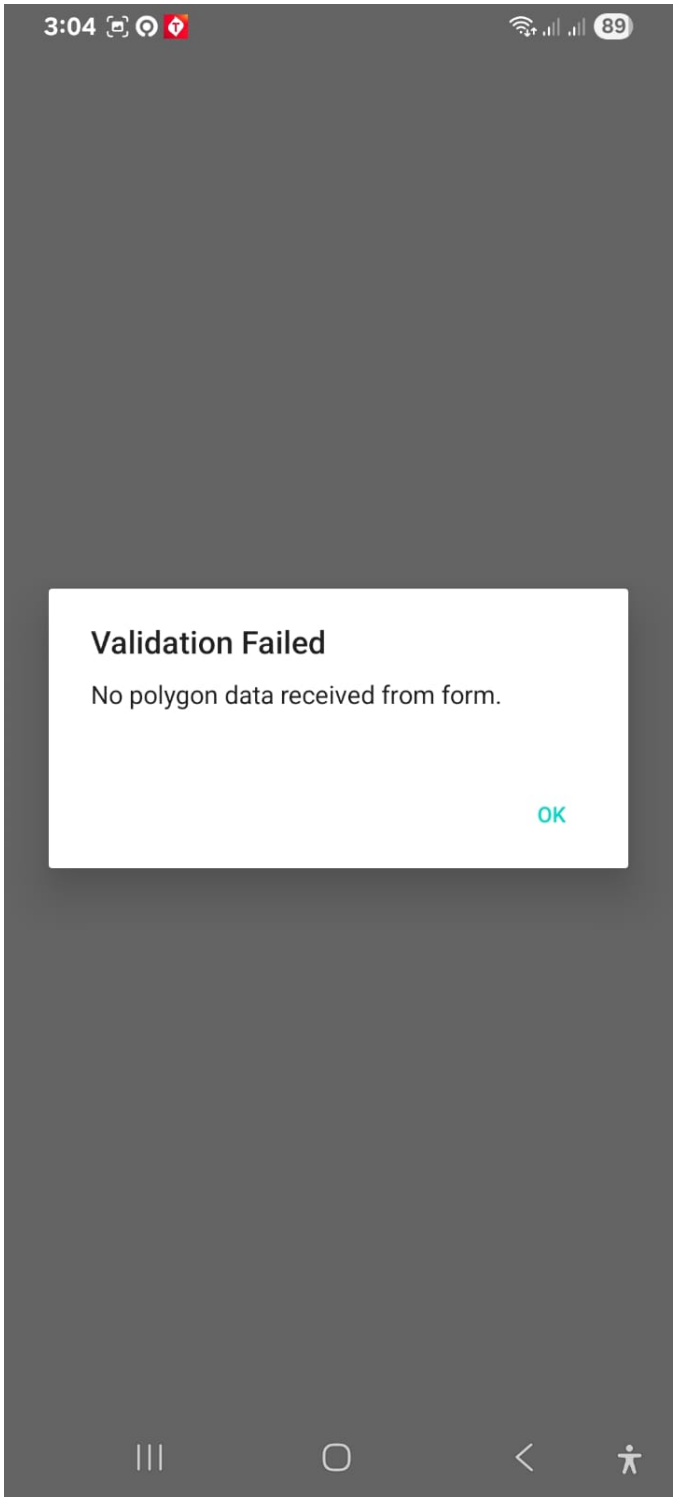
The following describes various validation scenarios and error messages related to plot/boundary recording in forms:

ODK External Validation Screen	Description



GV.1 - Required Field Error

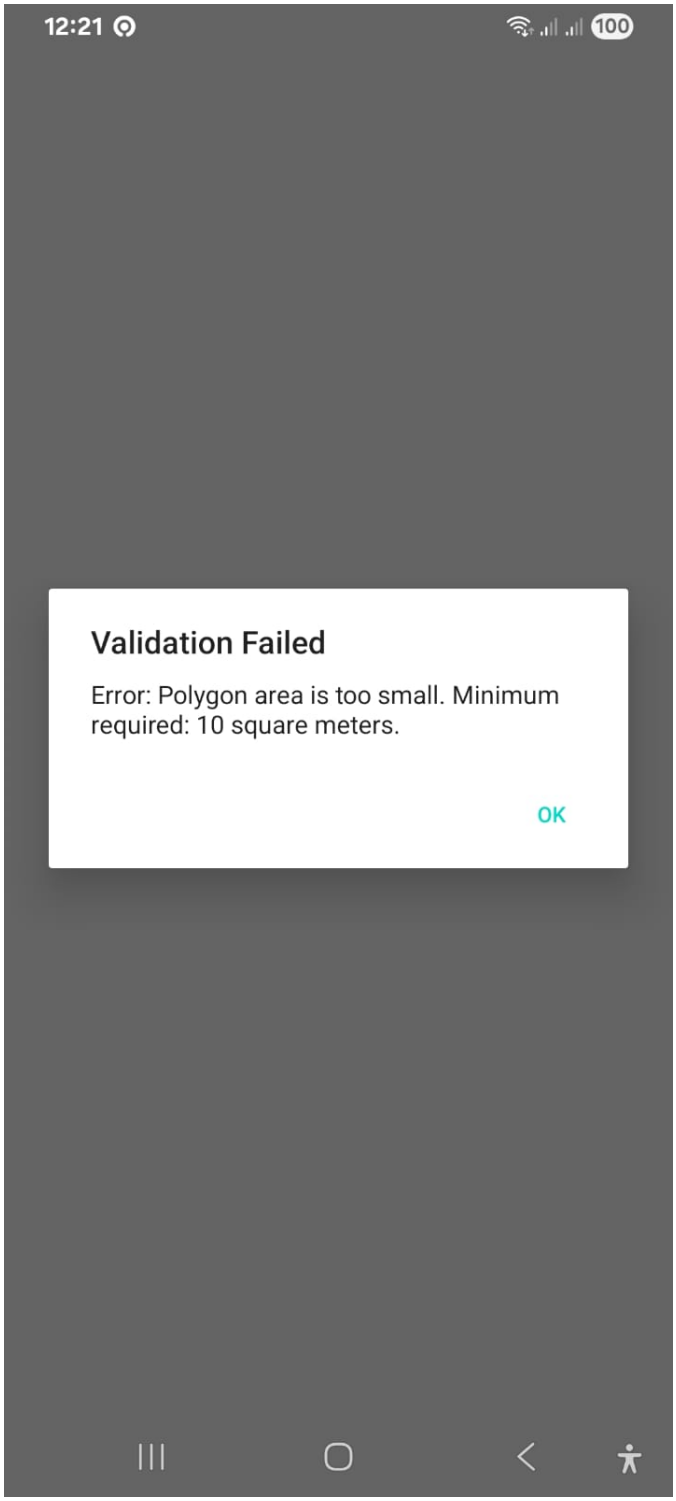
The validation field for the plot is mandatory. When the user presses the continue button without taking any action (such as pressing the launch button to record a plot), an error message will appear: "*Sorry, this response is required!*"



GV.2 - Empty or Null Plot Error

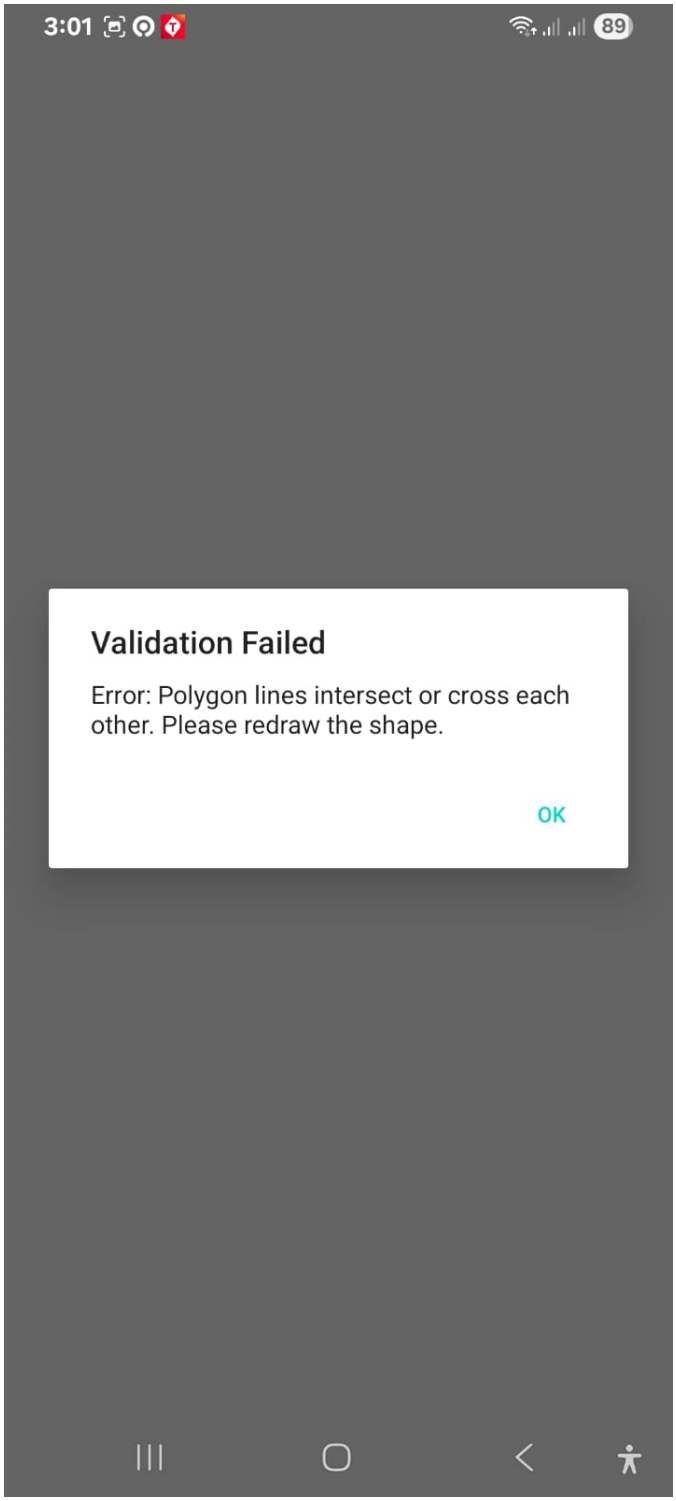
This error message appears if:

- The recorded plot/boundary is empty, OR
- The polygon field is not set as mandatory and results in a null value



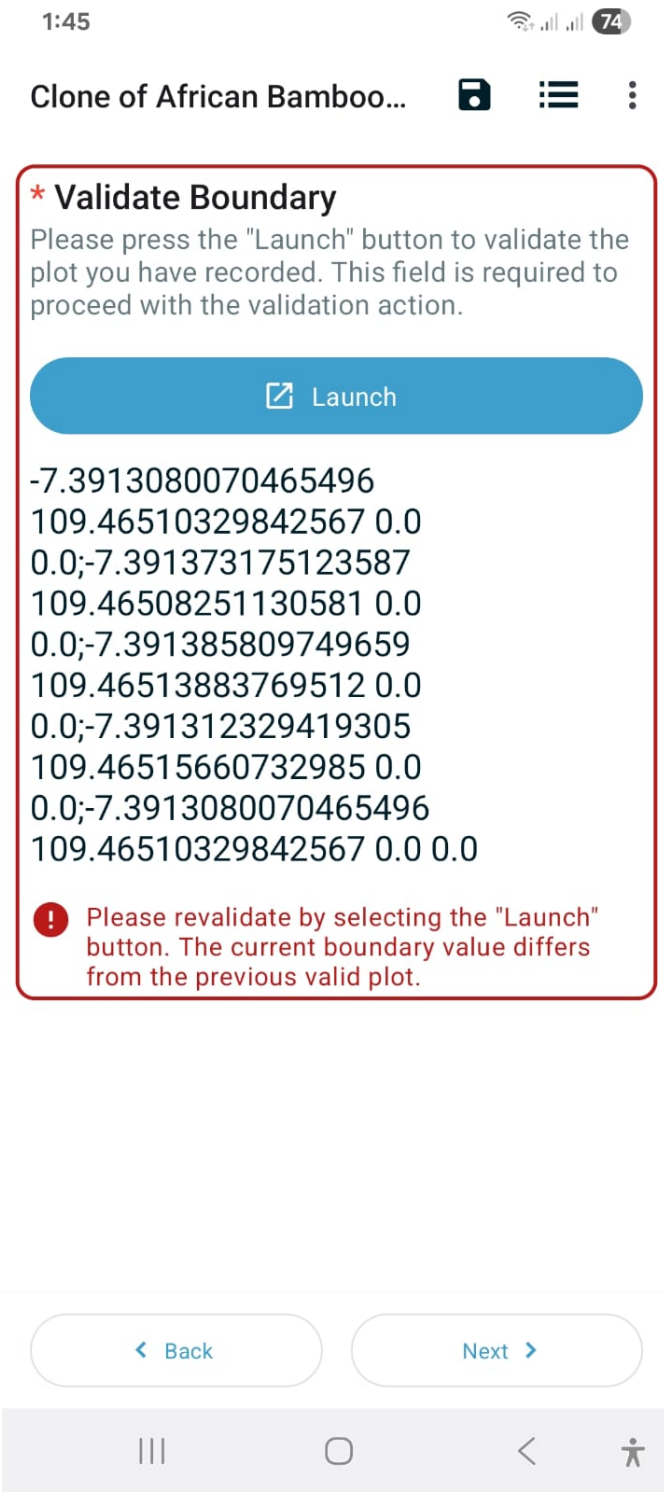
GV.3 - Minimum Area Error

This error message appears if the drawn/recorded plot is too small, with an area of less than **10 square meters**.



GV.4 - Invalid Geometry Error

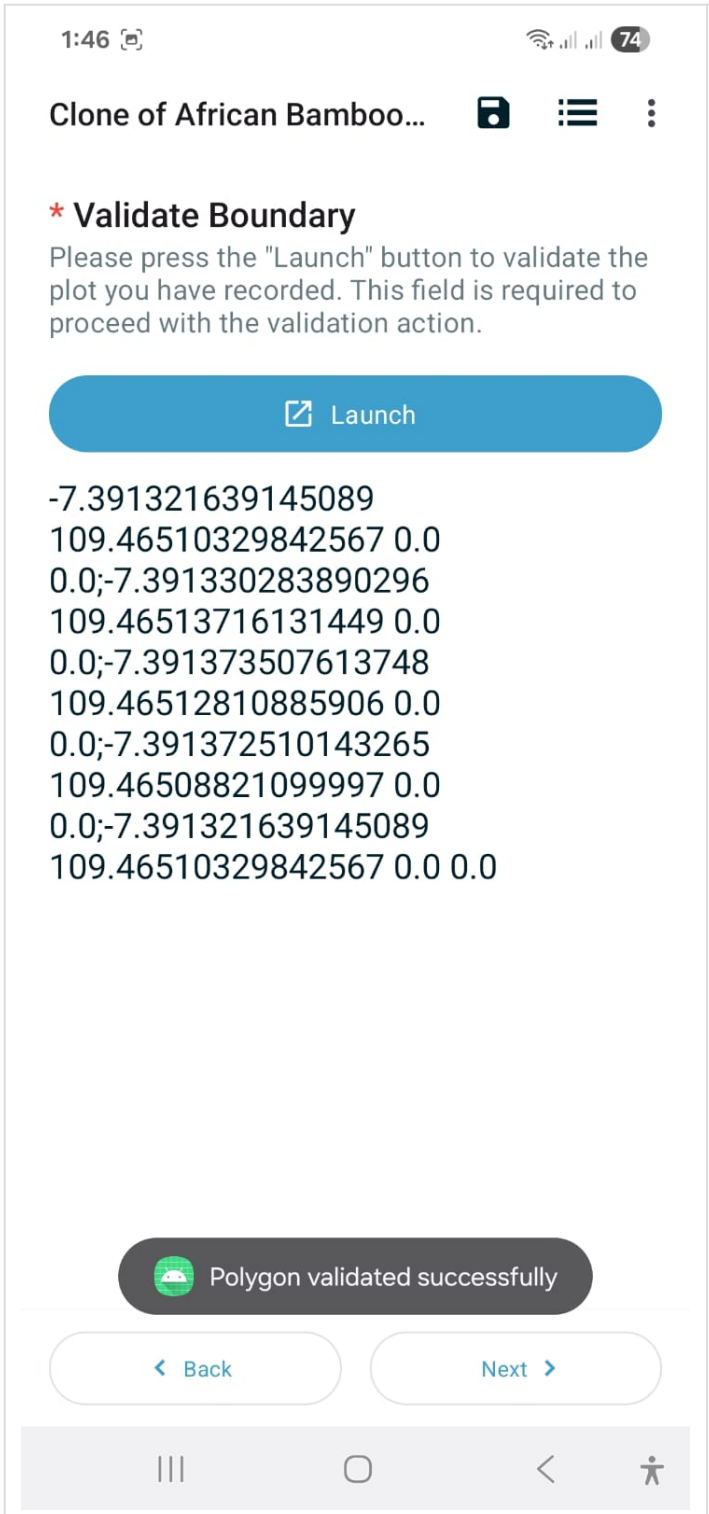
This error message appears if the drawn/recorded plot/boundary has **overlapping lines** or self-intersecting geometry, which does not meet valid polygon criteria.



GV.5 - Re-validation Required Error

This error message appears when:

- The user has already validated a valid plot/boundary
- They navigate back to a previous question to re-record the plot/boundary
- Validation needs to be performed again to ensure the plot remains valid



GV.6 - Successful Validation

This is an example of a successful validation response. The plot/boundary value will be automatically copied to the validation field, preserving it for future reference (such as in the GV.5 re-validation scenario).

Application Development

1. Overview

The African Bamboo - ODK External Validations is an Android client for KoboToolbox API integration. It downloads form submissions, stores them locally, and supports incremental sync for offline access.

Implementation Progress

Component	Status	Notes
Authentication	✅ Complete	Encrypted storage with BasicAuth
API Integration	✅ Complete	Pagination, delta sync
Database	✅ Complete	Room with hybrid schema
All 6 Screens	✅ Complete	Jetpack Compose + Material 3
Navigation	✅ Complete	Type-safe routes
ViewModels	✅ Complete	StateFlow/MVVM
Geoshape Display	⏳ Planned	Map thumbnails not yet implemented
Testing	⚠️ Minimal	Test skeletons only

---

2. API Integration

Endpoint

```
GET /api/v2/assets/{asset_uid}/data.json
```

Authentication

Implementation: `BasicAuthInterceptor.kt`

- Uses HTTP Basic Auth via OkHttp Interceptor
- Credentials stored in `EncryptedSharedPreferences`
- Dynamic base URL support for custom Kobo instances

Pagination

Implementation: `KoboRepository.kt`

- Page size: `limit=10`
- Cursor-based: follows `next` URL until null
- All pages inserted within single transaction

Delta Sync (Resync)

Query Parameter:

```
query={"_submission_time": {"$gte": "{last_sync_timestamp}"}}
```

- Timestamp format: ISO 8601 (UTC)
- Stored in `FormMetadataEntity.lastSyncTimestamp`

---

3. Data Model & Database Schema

Architecture Decision: Hybrid Storage

Instead of form-specific columns, we use:

- **Typed columns** for system fields (`_uuid`, `submissionTime`, `submittedBy`)
- **JSON blob** (`rawData`) for all form answers

This enables support for any KoboToolbox form without schema changes.

Entity: `SubmissionEntity`

File: `data/entity/SubmissionEntity.kt`

Field	Type	Source	Description
`_uuid`	String (PK)	`_uuid`	Unique submission identifier
`assetUid`	String	Form ID	Links submission to form
`_id`	Int	`_id`	Kobo's internal ID
`submissionTime`	Long	`_submission_time`	Parsed ISO 8601 → timestamp
`submittedBy`	String?	`_submitted_by`	Username of submitter
`instanceName`	String?	`meta/instanceName`	Form instance name
`rawData`	String	Full JSON	Complete submission payload
`systemData`	String?	Extracted	Geolocation, tags, etc.

Indices: `assetUid`, `submissionTime`, `instanceName`, `_uuid`

Entity: `FormMetadataEntity`

File: `data/entity/FormMetadataEntity.kt`

Field	Type	Description
`assetUid`	String (PK)	Form identifier

Field	Type	Description
`lastSyncTimestamp`	Long	Last successful sync time

DAOs

SubmissionDao (`data/dao/SubmissionDao.kt`)

- `insertAll()`, `insert()` - Bulk/single insert with REPLACE
- `getSubmissions(assetUid): Flow<List>` - Reactive list
- `getByUuid(uuid)` - Single submission lookup
- `getCount(assetUid)` - Total count
- `getLatestSubmissionTime(assetUid)` - For display
- `deleteByAssetUid()`, `deleteAll()` - Cleanup

FormMetadataDao (`data/dao/FormMetadataDao.kt`)

- `insertOrUpdate()` - Upsert pattern
- `getLastSyncTimestamp(assetUid)` - For delta sync

---

4. Screen Specifications

Screen 1: Login

File: `ui/screen/LoginScreen.kt` **ViewModel:** `LoginViewModel.kt`

Element	Implementation
Username field	Text input, required
Password field	Password input, obscured
Server URL	Text, default: `https://eu.kobotoolbox.org`
Form ID	Text, required
Submit button	"Download Data"

Workflow:

1. Validate all fields non-empty
2. Save credentials to `EncryptedSharedPreferences`
3. Navigate to Loading screen (DOWNLOAD type)

Note: Original spec called for `/token` endpoint. Implementation uses BasicAuth directly—no token exchange needed.

Screen 2: Download Loading

File: `ui/screen/LoadingScreen.kt` **ViewModel:** `LoadingViewModel.kt`

Element	Implementation
Spinner	CircularProgressIndicator
Message	"Downloading data..."
Error state	Retry + Back to Login buttons

Logic:

1. Construct URL: `{server_url}/api/v2/assets/{form_id}/data.json`
2. Fetch all pages (limit=10, follow `next`)
3. Transform JSON → `SubmissionEntity` list
4. Batch insert to Room
5. Store `lastSyncTimestamp`
6. Navigate to Download Complete with metrics

Screen 3: Download Complete

File: `ui/screen/DownloadCompleteScreen.kt`

Element	Implementation
Total entries	Passed via navigation args
Latest submission	Formatted date string
"View Data" button	Navigate to Home
"Resync Data" button	Navigate to Loading (RESYNC)

Screen 4: Home/Dashboard

File: `ui/screen/HomeDashboardScreen.kt` **ViewModel:** `HomeViewModel.kt`

Element	Implementation
Submission list	`LazyColumn` with `Flow<List>`
Search	TopAppBar search field
Sort	Bottom sheet (Name A-Z/Z-A, Date New/Old)
Resync FAB	Navigate to Loading (RESYNC)
Logout	Menu option with confirmation
Item click	Navigate to Submission Detail

Geoshape Indicator: □ Not yet implemented. LLD specified map thumbnails or "Map Available" badge.

Screen 5: Resync Loading

File: `ui/screen/LoadingScreen.kt` (shared with Download)

Element	Implementation
Spinner	CircularProgressIndicator
Message	"Syncing data..."

Delta Sync Logic:

1. Get `lastSyncTimestamp` from `FormMetadataDao`
2. API call with `query` parameter (dates >= timestamp)
3. Diff against local:
   - **Added:** `_uuid` not in Room → Insert
   - **Updated:** `_uuid` exists AND remote `submissionTime` > local → Update
4. Count added/updated records
5. Update `lastSyncTimestamp` to now
6. Navigate to Sync Complete with counts

Screen 6: Sync Complete

File: `ui/screen/SyncCompleteScreen.kt`

Element	Implementation
Added records	Count from sync
Updated records	Count from sync
Latest sync time	Current timestamp
"Return to Dashboard"	Navigate to Home

Screen 7: Submission Detail

File: `ui/screen/SubmissionDetailScreen.kt` **ViewModel:** `SubmissionDetailViewModel.kt`

Element	Implementation
Title	Instance name or formatted date
Submitted on	Formatted timestamp
Submitted by	Username
Answers list	Parsed from `rawData` JSON

Answer Parsing:

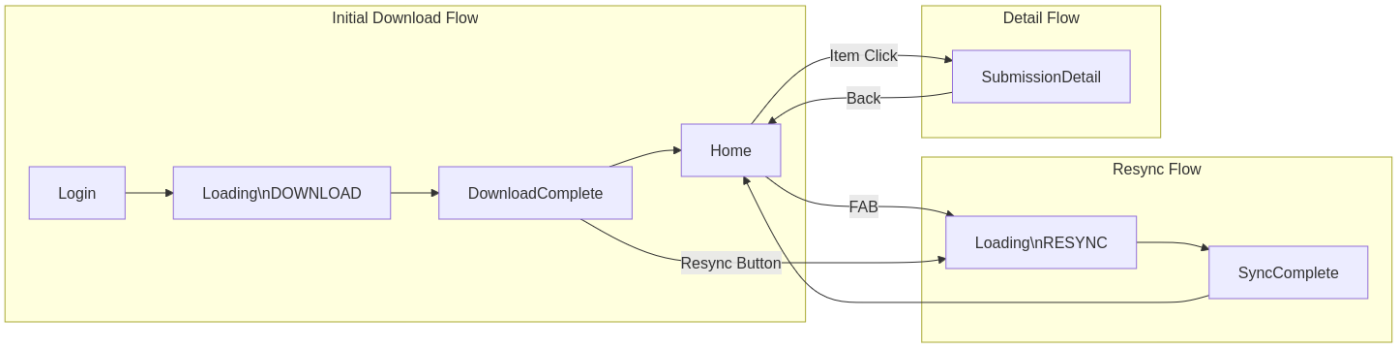
- Filters out system fields (prefix `_`, `meta`, `formhub`)

- Converts `snake_case` keys to `Title Case` labels
- Handles all JSON types (primitives, arrays, objects)

---

5. Navigation

File: `navigation/Routes.kt`, `navigation/AppNavHost.kt`



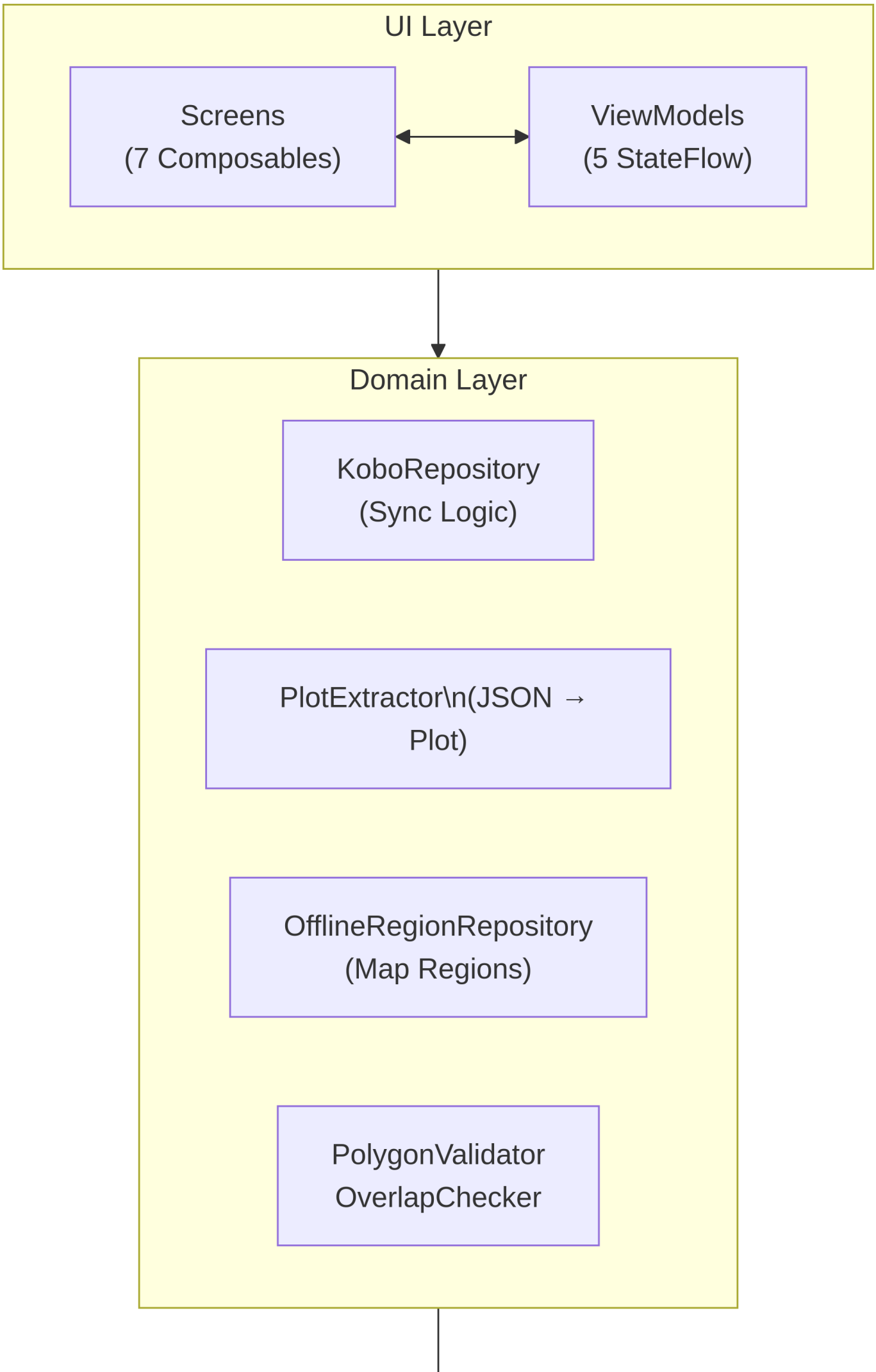
Route Definitions:

- `Login` - Entry point for logged-out users
- `Loading(type: LoadingType)` - DOWNLOAD or RESYNC
- `DownloadComplete(totalEntries, latestSubmissionDate)`
- `Home` - Entry point for logged-in users
- `SubmissionDetail(uuid)`
- `SyncComplete(addedRecords, updatedRecords, latestRecordTimestamp)`

---

6. Architecture

Pattern: MVVM + Clean Architecture


UI Layer
Screens
(7 Composables)
ViewModels
(5 StateFlow)
Domain Layer
KoboRepository
(Sync Logic)
PlotExtractor\n(JSON →
Plot)
OfflineRegionRepository
(Map Regions)
PolygonValidator
OverlapChecker

Dependency Injection

Framework: Hilt

Module	Provides
`NetworkModule`	`KoboApiService`, `OkHttpClient`, `Json`
`DatabaseModule`	`AppDatabase`, DAOs

State Management

- **ViewModels:** `StateFlow` for UI state
- **Database:** `Flow<List<T>>` for reactive lists
- **Collection:** `collectAsStateWithLifecycle()` in Composables

---

7. Security

Credential Storage

File: `data/session/SessionManager.kt`

- `EncryptedSharedPreferences` with AES256_GCM
- `MasterKey` with AES256_GCM scheme
- Stores: username, password, serverUrl, assetUid

Network Security

- BasicAuth over HTTPS only
- No token stored (credentials used per-request)
- Dynamic base URL validated before use

---

8. Definition of Done

Requirement	Status	Evidence
MVVM with StateFlow		All 4 ViewModels use `MutableStateFlow`
Jetpack Compose UI		All screens in `ui/screen/`
`collectAsStateWithLifecycle`		Used in all screen Composables
Login + encrypted session		`SessionManager` with AES256
Pagination (no missing records)		`KoboRepository.fetchSubmissions()` follows `next`

Requirement	Status	Evidence
Delta sync (Added vs Updated)	✅	`KoboRepository.resync()` with diffing
Offline access (Room cache)	✅	`SubmissionDao.getSubmissions()` returns cached data
Password/token encrypted	✅	`EncryptedSharedPreferences`

---

9. Known Gaps & Future Work

Not Yet Implemented

Feature	Priority	Notes
Geoshape map display	Medium	Show map thumbnail in list items
Geolocation validation	Low	External validation logic
Database migrations	High	Currently uses destructive migration
Comprehensive tests	High	Only skeleton files exist
Offline-first indicators	Low	No explicit offline mode UI
Error logging/Crashlytics	Medium	No crash reporting

Technical Debt

1. **Destructive migrations:** `fallbackToDestructiveMigration()` will lose data on schema changes
2. **Test coverage:** Minimal unit/instrumented tests
3. **Accessibility:** Limited `contentDescription` coverage

---

10. File Structure

```
app/src/main/java/com/akvo/externalodk/
├── data/
│   ├── dao/              # SubmissionDao, FormMetadataDao
│   ├── database/         # AppDatabase
│   ├── dto/              # KoboSubmissionDto, KoboDataResponse
│   ├── entity/           # SubmissionEntity, FormMetadataEntity
│   ├── network/          # KoboApiService, Interceptors
│   ├── repository/       # KoboRepository
│   └── session/          # SessionManager, AuthCredentials
```

```
├── di/                    # Hilt modules
├── navigation/            # Routes, AppNavHost
└── ui/
    ├── screen/            # All 6 Composable screens
    ├── theme/             # Material 3 theming
    └── viewmodel/         # Login, Loading, Home, SubmissionDetail
```

Appendix: API Response Example


```
{
  "count": 1250,
  "next": "https://kf.kobotoolbox.org/api/v2/assets/xxx/data.json?start=300",
  "previous": null,
  "results": [
    {
      "_id": 12345,
      "_uuid": "abc-123-def",
      "_submission_time": "2026-01-15T10:30:00Z",
      "_submitted_by": "field_worker",
      "meta/instanceName": "Survey 001",
      "question_1": "Answer 1",
      "question_2": 42,
      "geoshape": "..."
    }
  ]
}
```