SaudaPakka KYC

Frontend Integration Guide

This guide details how to integrate the Aadhaar-based KYC flow into the SaudaPakka web/mobile application. The process uses a Redirect + Polling architecture.

The KYC Workflow

1 **Initiate**

Frontend calls our API to get a unique government-hosted DigiLocker URL.

2 **Consent**

User is redirected to DigiLocker (Government site) to enter Aadhaar and OTP.

3 **Callback**

User is redirected back to our app with a `session_id`.

4 **Verification**

Frontend calls our backend to "Fetch and Save" the official data.

🚀 API Endpoints

1. Initiate KYC Session

Endpoint: `POST /api/kyc/initiate/`

Auth Required: `Bearer <JWT_TOKEN>`

Request Body:

```json
JSON

{
    "redirect_url": "http://localhost:3000/kyc/callback"
}
```

Note: The `redirect_url` is where the government will send the user after they finish.

Success Response (200 OK):

```json
JSON

{
    "digilocker_url": "https://digilocker.meripehchaan.gov.in/...",
    "entity_id": "339da5d5-b0f9-4a32-a819-5bc75d2ef00f"
}
```

2. Verify & Fetch Status

Endpoint: `POST /api/kyc/verify-status/`

Auth Required: `Bearer <JWT_TOKEN>`

Request Body:

```
JSON
```

```
{
    "entity_id": "339da5d5-b0f9-4a32-a819-5bc75d2ef00f"
}
```

Responses:

● Success (200 OK) **Data has been saved to the DB.**

```json
{
    "status": "SUCCESS",
    "message": "Identity Verified Successfully!",
    "data": { "name": "RAJESH KUMAR" }
}
```

● Processing (202 Accepted) **User authorized, but government servers are still sending data.**

```json
{
    "status": "PROCESSING",
    "message": "User authorized, but data is still being fetched. Retry in 3s."
}
```

Action: Frontend should wait 3 seconds and call this endpoint again.

💻 Frontend Implementation Logic (Next.js/React)

Step A: The Initiation Page

When the user clicks "Verify with Aadhaar":

```javascript
const handleStartKYC = async () => {
    const res = await api.post('/kyc/initiate/', {
        redirect_url: `${window.location.origin}/kyc/callback`
    });
    // Save entity_id locally if needed, then redirect
    window.location.href = res.data.digilocker_url;
};
```

Step B: The Callback Page (`/kyc/callback`)

This page handles the "Return" from the government. It should show a loading spinner while polling our backend.

```javascript
// Inside useEffect or a handler on the callback page
const verifyIdentity = async (entity_id) => {
    try {
        const res = await api.post('/kyc/verify-status/', { entity_id });

        if (res.status === 200) {
            router.push('/dashboard?kyc=success');
        } else if (res.status === 202) {
            // Data is still processing at Sandbox/Govt level
            setTimeout(() => verifyIdentity(entity_id), 3000); // Poll every 3
seconds
        }
    } catch (err) {
        router.push('/kyc/error');
    }
};
```

⚠️ Important Notes for Frontend Team

URL Search Params:

On the `/kyc/callback` page, the government appends the `entity_id` to the URL. Use `new URLSearchParams(window.location.search).get('entity_id')` to retrieve it.

User Persistence:

Ensure the user remains logged in during the redirect. The JWT token should be stored in `localStorage` or a `HttpOnly Cookie`.

State Management:

Once the `verify-status` returns `SUCCESS`, the global `user.is_kyc_verified` state should be updated to unlock Seller/Broker features.

SaudaPakka KYC Frontend Integration Guide

Aadhaar-based KYC flow with Redirect + Polling architecture