Most Important Client Code for AWS API Calls — How It Works

This paper explains the React-side logic that talks to your AWS backend. It covers authentication token handling, list/display, download, and both single-part and multipart uploads using presigned S3 URLs obtained through your API.

1) Authentication & Routing

The app mounts an OpenID Connect provider at startup and redirects the user to /app after sign-in. Private routes (e.g., /app/files) are gated by isAuthenticated. When the user logs out, the Hosted UI /logout endpoint is used.

2) API Client Responsibilities

Client helpers (apiGet/apiPost) send requests to API Gateway endpoints and attach the user’s ID or access token in the Authorization header. Endpoints include /files/list, /downloads/create, /uploads/create, and the multipart trio: /uploads/multipart/create, /uploads/multipart/part-url, /uploads/multipart/complete.

3) Listing Documents

On the Files page, loadList() calls /files/list with the JWT. Results populate the table. A 401 triggers a sign-in redirect.

4) Downloading Documents

handleDownload() calls /downloads/create with the key. The backend authorizes the request and returns a short-lived presigned GET URL. The browser opens that URL directly against S3; the API never streams the bytes.

5) Single■Part Upload Flow

For small files, uploadOne() asks the API for a presigned PUT URL via /uploads/create and then issues a single XMLHttpRequest PUT to S3 with Content-Type set. Progress events are used to update the UI, and status becomes 'done' when S3 returns 2xx.

6) Multipart Upload Flow (Large Files)

If file.size ≥ threshold, the client uses S3 multipart upload via three API calls: (1) create (gets uploadId), (2) part-url (presigned PUT URL per part), (3) complete (commits parts with their ETags). Each part is uploaded with its own XMLHttpRequest PUT. Importantly, the client does not set Content-Type on part PUTs (as it would break the signature). A small worker pool uploads several parts concurrently and updates progress based on bytes uploaded. After all parts succeed, the client calls complete with the list of {PartNumber, ETag}.

7) Security & Reliability Rationale

JWTs protect the control plane (API Gateway). Presigned URLs protect the data plane (S3). Short expirations and method scoping ensure least-privilege writes/reads. 401s trigger re-auth. Multipart upload concurrency is bounded, and errors bubble to the UI with an 'error' status. After uploads complete, the list view refreshes to reflect the latest DynamoDB state.

This client-side design keeps large data transfers off the API, uses short■lived credentials for S3, and scales cleanly as files grow or concurrency increases.