# 0) Executive Summary

We added a third upload path in AgentFleet that lets users sign in with Microsoft (Entra ID) and choose files from SharePoint/OneDrive they already have access to. The solution uses delegated auth in the browser (MSAL, PKCE), optional server‑side On‑Behalf‑Of (OBO) for ingestion, and a Graph‑powered Explorer (Sites → Libraries → Folders → Files). A client‑side **Download Selected** flow is included.

# 1) Approach Documentation

## 1.1 Objectives

* Allow end users to authenticate with Microsoft in one click.
* Browse user‑accessible SharePoint **Sites → Document Libraries → Folders → Files**.
* Pick files to either (a) download locally in the browser, or (b) upload to AgentFleet via server‑side ingestion.
* Enforce least‑privilege, user‑scoped access (delegated), no tenant‑wide app‑only permission.

## 1.2 Architecture Overview

* **Frontend (SPA, React + MSAL)**: Auth Code + PKCE; obtains delegated tokens with scopes openid profile email offline\_access Files.Read and Sites.Read.All (for site/library enumeration). Provides an Explorer UI and a local **Download Selected** feature.
* **Backend (Node/Express + MSAL‑Node)**: Confidential client with client secret. Performs **OBO** using the user’s token to call Graph and stream file bytes to AgentFleet ingestion. (Optional when only local download is required.)
* **AgentFleet Ingestion (existing)**: Receives streamed files and metadata. In dev we used an ingestion mock.

## 1.3 Key Design Choices

* **Two app registrations**: SPA (public client) for browser login; Backend (confidential) for OBO. Clear secret boundary and principle of least privilege.
* **Delegated over app‑only**: Users only see content they’re permitted to. No Sites.Selected or app‑wide read needed.
* **Graph Explorer vs Picker**: Implemented a native Graph Explorer for deterministic Sites→Libraries→Folders navigation. (We can still add Microsoft File Picker v8 as an alternative.)
* **By‑path site resolution**: Normalize deep page/file URLs to site roots and resolve via GET /v1.0/sites/{host}:/sites/{name} for reliability.

## 1.4 Non‑Goals / Out of Scope

* Write‑back to SharePoint (upload/update files).
* Enterprise DLP/virus scanning (hook points are documented in Implementation).
* Cross‑tenant app‑only crawling.

## 1.5 Security & Compliance

* SPA uses PKCE; tokens stored in session storage only.
* Backend keeps secrets server‑side; OBO tokens not exposed to browser.
* All access is end‑user delegated; audit trail available (who picked, from which site/drive/item, when).
* CORS restricted to AgentFleet domains; secrets in vault for prod.

## 1.6 Success Criteria

* User can authenticate with Microsoft and see their Sites → Libraries → Folders.
* User can pick files and either (a) **Download Selected** locally, or (b) send to ingestion (server OBO path) with < 3 clicks after selection.
* No permission elevation beyond user’s rights; no secrets in the browser; no tenant‑wide app‑only scope.

## 1.7 Risks & Mitigations

* **Consent friction**: Use admin consent for SPA scopes; surface helpful error states.
* **Large folders/files**: Virtualized lists; streaming download via downloadUrl; retry/backoff on network errors.
* **Rate limits (Graph)**: Exponential backoff; concurrency caps.
* **Multi‑tenant SaaS**: Ensure SPA app is multi‑tenant; implement per‑tenant consent flow and issuer validation.

# 2) Implementation Documentation

## 2.1 Repositories & Structure

* **frontend/**: React (Vite) + MSAL. Components:
  + msal.js (sign‑in, token, sign‑out)
  + msgraph.js (helpers: sites by path, drives, children, downloadUrl)
  + Explorer.jsx (Sites → Libraries → Folders → Files, selection)
  + App.jsx (tabs, Microsoft flow, Download Selected)
* **backend/**: Node/Express + MSAL‑Node.
  + src/obo.js (OBO exchange)
  + src/graph.js (item metadata + stream via downloadUrl)
  + src/index.js (routes: /health, /api/ms/download-and-forward)
* **ingestion-mock/**: Simple receiver to validate ingest.

## 2.2 App Registrations (Entra ID)

* **SPA App** (public): Redirect URIs include http://localhost:5173 and prod. Enable ID & Access tokens. Delegated scopes: openid profile email offline\_access Files.Read Sites.Read.All. Admin consent recommended.
* **Backend App** (confidential): Client ID + Secret. No Graph app permissions required for OBO; uses /.default when exchanging on behalf of the user.

## 2.3 Environment Variables

* **frontend/.env**
  + VITE\_AZURE\_AD\_TENANT\_ID=<tenant id or 'common'>
  + VITE\_AZURE\_AD\_CLIENT\_ID=<SPA client id>
  + VITE\_GRAPH\_SCOPES="openid profile email offline\_access Files.Read Sites.Read.All"
  + VITE\_PICKER\_BASEURL\_SHAREPOINT=https://<tenant>.sharepoint.com
  + VITE\_PICKER\_BASEURL\_ONEDRIVE=https://<tenant>-my.sharepoint.com
  + VITE\_BACKEND\_URL=http://localhost:4000
* **backend/.env**
  + TENANT\_ID=<tenant id>
  + BACKEND\_CLIENT\_ID=<confidential app id>
  + BACKEND\_CLIENT\_SECRET=<secret>
  + BACKEND\_REDIRECT\_URI=http://localhost:4000/auth/redirect
  + INGEST\_URL=http://localhost:5001/ingest
  + PORT=4000

## 2.4 Flows

**A) Client‑side Download** 1. SPA gets token (delegated). 2. For each selected item {driveId, id} → GET /drives/{driveId}/items/{id}?$select=name,@microsoft.graph.downloadUrl. 3. Browser downloads either by direct link (lightweight) or by fetching blob and saving with desired filename.

**B) Server‑side Ingestion (optional)** 1. SPA sends selected items to backend with Authorization: Bearer <user token>. 2. Backend performs OBO → gets Graph token. 3. Backend streams bytes from downloadUrl and forwards to ingestion endpoint.

## 2.5 UI/UX

* Tabs: **Direct Upload | S3 Upload | Microsoft**; top‑right shows **Sign in / Sign out**.
* Explorer: site chips (recent/followed) + site URL input; drives; breadcrumbed folders; file table with Pick/Unpick; actions (**Use selected**, **Download Selected**).

## 2.6 Error Handling

* Token acquisition: retry popup if silent fails; show clear error if consent missing.
* Graph calls: surface HTTP status and friendly messages; backoff on 429.
* Navigation: resilient to empty libraries; display “No items” states.

## 2.7 Observability

* Frontend: basic event logs (sign‑in, site chosen, drive chosen, download/ingest clicked).
* Backend: request logs, Graph call timing, ingestion response, error metrics (429/5xx).

## 2.8 Deployment Notes

* Lock CORS to AgentFleet origins.
* Secrets in vault; rotate quarterly.
* Use HTTPS everywhere; same‑site cookies if you add sessions.
* Multi‑tenant: verify issuer, store per‑tenant consent/metadata.

## 2.9 Rollback Plan

* Feature flag the Microsoft tab.
* Backend route can be toggled off independently.
* Revert to Direct/S3 only if outages occur.

# 3) Use Cases Documentation

## 3.1 Primary Use Cases

* **UC‑1: Browse & Download** (Delegated, client‑side)
  1. User signs in with Microsoft.
  2. User selects a site → document library → folder → file(s).
  3. Clicks **Download Selected** → files saved locally.
  4. Acceptance: Files download successfully with correct names and sizes.
* **UC‑2: Browse & Ingest to AgentFleet** (Delegated + OBO)
  1. User signs in with Microsoft.
  2. Selects files via Explorer.
  3. Clicks **Upload Selected** → backend OBO downloads and forwards to ingestion.
  4. Acceptance: Ingestion receives files; pipeline begins processing; audit info recorded.
* **UC‑3: Least‑Privilege Access**
  1. User only sees sites/libraries and files they already have permission to.
  2. Acceptance: A user without access cannot view or ingest restricted content.
* **UC‑4: Sign Out**
  1. User clicks **Sign out** tab.
  2. Acceptance: MSAL session cleared; protected actions require re‑auth.

## 3.2 Edge Use Cases

* **UC‑E1: Very Large Folder**
  + Expected: Paging/virtualization; Graph throttling respected; user is informed.
* **UC‑E2: Large Files**
  + Expected: Direct downloadUrl works; server ingest streams without buffering entire file; progress/timeout handling.
* **UC‑E3: Missing Scopes/Consent**
  + Expected: Friendly error; sign‑out/in re‑prompts; admin consent guidance shown.
* **UC‑E4: Multi‑tenant Customers**
  + Expected: SPA supports common authority; per‑tenant admin consent captured; issuer validated on backend.

## 3.3 Non‑Functional Requirements

* **Performance**: Initial site list < 2s (cached recent/followed); folder listing < 1s p50.
* **Reliability**: Graceful retries on Graph 429/5xx; user‑visible error messages.
* **Security**: No secrets in frontend; OBO tokens short‑lived; scopes least‑privilege.

# 4) Jira Backlog (suggested)

* **AFE‑MS‑001**: SPA App registration (multi‑tenant), configure scopes & redirects.
* **AFE‑MS‑002**: Backend App registration (confidential), secret in vault.
* **AFE‑MS‑003**: MSAL integration (sign‑in, silent token, sign‑out).
* **AFE‑MS‑004**: Explorer (sites → drives → folders → files) with by‑path site resolution.
* **AFE‑MS‑005**: Client‑side Download Selected (downloadUrl + blob).
* **AFE‑MS‑006**: Backend OBO endpoint /api/ms/download-and-forward + streaming ingest.
* **AFE‑MS‑007**: CORS, telemetry, error surfaces.
* **AFE‑MS‑008**: E2E QA matrix (tenants, roles, large files, throttling).
* **AFE‑MS‑009**: Production rollout & feature flag.

# 5) Acceptance Checklist

* User can sign in/out with Microsoft.
* Sites list appears (recent/followed) and accepts pasted site URLs.
* Libraries list for a chosen site; folders/files navigate with breadcrumbs.
* **Download Selected** saves files locally.
* (Optional) **Upload Selected** streams to AgentFleet ingestion via backend OBO.
* No elevated permissions beyond user’s delegated access.
* Errors are user‑readable; logs/metrics captured.