**Detailed Explanation of Each Subscriber Use Case**

Let's walk through each Subscriber use case in the RW Tool project. For each one, we'll cover the main steps, decisions, and expected outcomes. This will give you a solid understanding of how a subscriber (user) interacts with the system at every stage.

**UC-S1: Sign Up & Sign In**

**Goal:** Allow users to create an account and access the RW Tool.

* **Step 1:** The user visits the sign up or sign in page from the web app.
* **Step 2:** For sign up, they enter a valid email and strong password (or use SSO if available).
* **Step 3:** The system validates credentials. If registering, an account is created (with optional email verification).
* **Step 4:** The user submits their credentials to sign in. Passwords are checked securely (hashed comparison).
* **Decisions and Outcomes:**
  + If credentials are correct and the account is active, the system returns a JWT session token and the user is authenticated.
  + If the credentials are incorrect, a 401 error is returned; if the account is locked, a 423 error may show.
* **Result:** The user gets access to the RW Tool with session credentials, or sees an error (wrong password, locked account).

**UC-S2: Request Subscription to Domain(s)**

**Goal:** Allow a subscriber to request access to specific report domains (e.g., "Equities").

* **Step 1:** After logging in, the user is shown a list of available domains.
* **Step 2:** The subscriber selects one or more domains they wish to subscribe to.
* **Step 3:** The frontend sends a POST /subscriptions request to the backend with the selected domain(s), creating a subscription in PENDING status.
* **Step 4:** The system checks if a subscription for that user+domain combination already exists. If so, it returns a 409 with existing status.
* **Decisions and Outcomes:**
  + If it’s new, the request is logged for admin review. If duplicate, the system advises the user.
* **Result:** The subscription is now awaiting admin approval, and the user gets confirmation their request was sent.

**UC-S3: View Subscription Status & Notifications**

**Goal:** Enable subscribers to check their subscription requests and get notified about decisions.

* **Step 1:** The user checks the status of their domain subscriptions on the portal (calls GET /subscriptions?me=1).
* **Step 2:** The system lists all the user's subscriptions with current status (PENDING, APPROVED, or REJECTED).
* **Step 3:** When an admin approves/rejects a request, the user gets an in-app notification and/or email with the result and reason (if rejected).
* **Result:** The subscriber always knows the current state of their requests and is notified of any changes.

**UC-S4: Browse Catalog**

**Goal:** Let a subscriber see and search reports relevant to their approved domains.

* **Step 1:** Once a user's subscription is APPROVED for a domain, they can access the report catalog.
* **Step 2:** The user can filter, search, and paginate reports, but only for domains where access is granted.
* **Step 3:** The system enforces access control on the backend—users only see reports for their allowed domains.
* **Edge:** If a user has no approved domains, the catalog displays a helpful empty state message.
* **Result:** Subscriber can view report lists with metadata (title, date, type) and stay within their permission scope.

**UC-S5: View Report (Watermarked)**

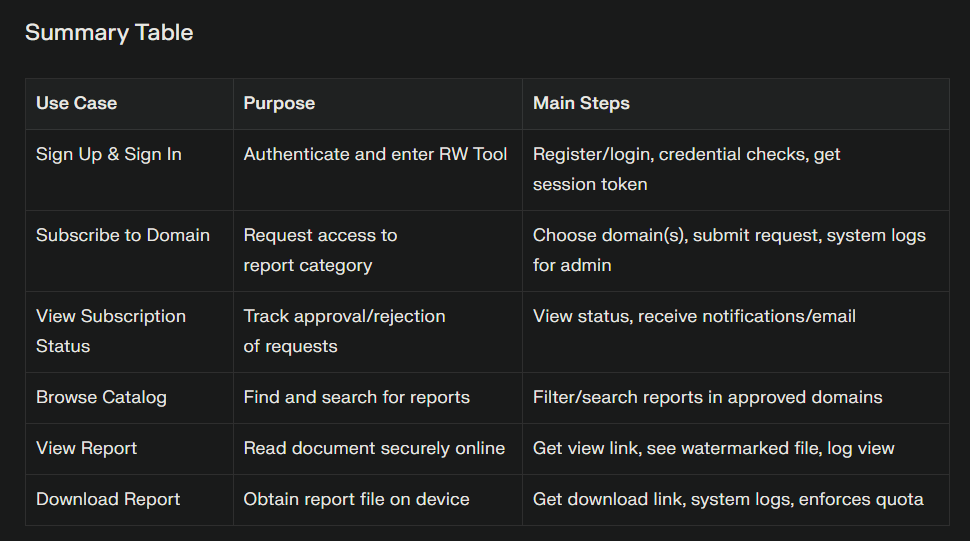
**Goal:** Allow users to view detailed reports securely within the web app, with their identity watermarked on the document.

* **Step 1:** Subscriber selects a report from the catalog to view.
* **Step 2:** The frontend requests a short-lived (pre-signed) URL for viewing from the backend (mode=view).
* **Step 3:** The app presents the report in an in-browser PDF viewer with a visible user ID and timestamp watermark overlay.
* **Step 4:** All views are audit-logged (but do **not** count as downloads).
* **Edge:** If the pre-signed link expires, the user is prompted to refresh; if access is revoked, they see a 403 error.
* **Result:** User reads the document without being able to remove attribution; all actions are tracked.

**UC-S6: Download Report**

**Goal:** Permit users to download reports for domains they're authorized for, while logging this access.

* **Step 1:** Subscriber clicks the download button on a report.
* **Step 2:** The app requests a short-lived pre-signed URL from the backend (mode=download).
* **Step 3:** If the user is within their download quota, and the file is not archived or restricted, the download proceeds.
* **Step 4:** A backend job logs the download (who, what, when) for auditing and compliance.
* **Edge:** If quota is exceeded, a 429 error is shown. If the report is archived, a 404 error is returned.
* **Result:** User obtains a secure file, audit logs are updated, system enforces quotas and states.



**Admin Use Cases**

**UC-A1: Approve/Reject Subscription Request**

**Goal:** To allow Admins to manage user access by approving or rejecting their domain subscription requests.

* The Admin logs into the Admin portal and sees a queue of all pending subscription requests.
* The Admin reviews each request, examines the user's details, requested domain, and any justification or context.
* The Admin chooses to approve (grant access) or reject (deny access) the request. Optionally, they add a reason for rejection.
* The system updates the subscription status, notifies the user (via email and in-app), and logs this decision for audit and compliance.
* If another Admin has already processed the request, the UI will show the latest status to avoid race conditions.
* **Result:** Subscription status is changed, the user is triggered for notification, and a full audit trail is created for compliance and traceability.

**UC-A2: Manage Domains**

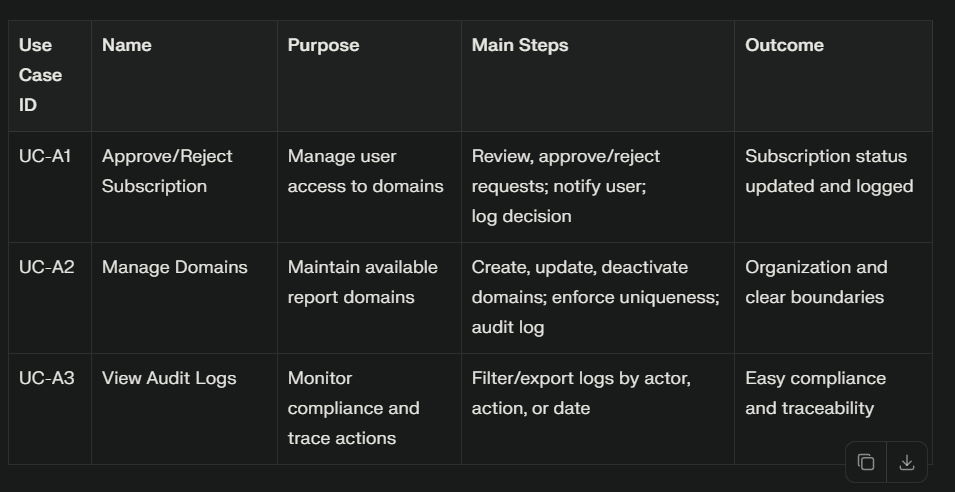
**Goal:** To maintain and organize the set of report domains available for subscription.

* The Admin accesses the domains management panel.
* Admins can create new domains, update names/descriptions, or (attempt to) delete domains.
* If a domain has reports attached, deletion is blocked (suggest “deactivate” domain instead to avoid data loss).
* All domain CRUD (create/read/update/delete) actions are logged for audit.
* **Result:** Domains remain unique and consistently maintained; taxonomy and access boundaries are clear.

**UC-A3: View Audit Logs**

**Goal:** To enable compliance and track user/admin actions across the RW Tool system.

* Admins use filters (actor, action, date) to search and export audit logs (CSV or similar).
* They may inspect who approved what, who accessed which data, and monitor unusual or privileged actions.
* Exported logs serve for audits, investigations, or compliance evidence.
* Audit logs are immutable and redact PII as needed.
* **Result:** Admins can present compliance reports and trace all changes and accesses within the system.



**Ops Use Cases**

**UC-O1: See New Ingestion Events**

**Goal:** To make incoming reports from external sources visible for review and processing.

* Ops users view a dashboard showing NEW ingestion events (triggered by webhook or polling external provider).
* Each event includes a preview of its payload: source, domain, metadata, and status.
* **Result:** Ops team is aware of all new content waiting for processing and can triage before syncing.

**UC-O2: Trigger Sync for Selected Events**

**Goal:** To fetch and process new reports, making them available internally.

* Ops selects one or more events and clicks “Sync.”
* This triggers background jobs (e.g., fetch file from remote, virus scan, watermark, upload to local storage, extract metadata).
* Job status is updated in real-time (QUEUED, DOWNLOADED, SCANNED, PUBLISHED, FAILED).
* If jobs fail (network error, file error), retries are attempted, and persistent failures go to the Dead Letter Queue (DLQ) for manual review.
* **Result:** Reports are ingested, verified, and made available; errors are handled cleanly.

**UC-O3: Handle Failures & Quarantine**

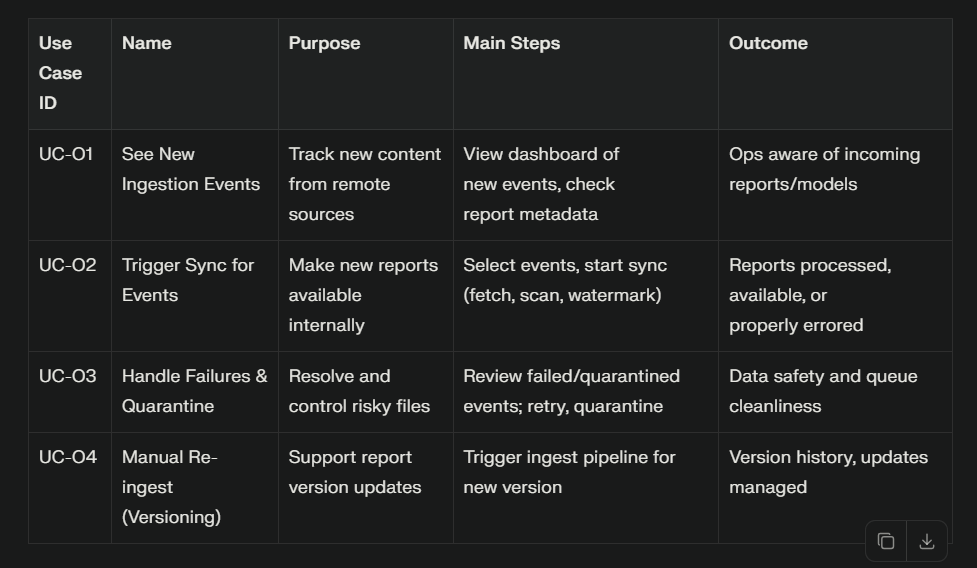
**Goal:** To identify, diagnose, and resolve ingestion problems while ensuring compliance.

* Ops sees FAILED or QUARANTINED events in their dashboard, with error details.
* They can retry or discard/reject failures.
* Quarantined items (e.g., virus-positive files) stay undiscoverable to users but visible to Ops/Admin for oversight.
* **Result:** Queue remains clean, compliance audit is possible, and risky files do not expose data.

**UC-O4: Manual Re-ingest (Versioning)**

**Goal:** To allow Ops to trigger refresh of reports when external updates or new versions are available.

* Ops provides the external source\_ref and indicates a new version.
* Ingestion pipeline repeats: file is fetched, checked, and a new version created under the same logical report.
* Past versions remain accessible (unless archived).
* **Result:** System supports versioned reporting and traceable updates for content history.



**System/Integration Use Cases**

**UC-SYS1: Receive Remote Webhook**

**Goal:** To integrate external storage providers with RW Tool using automated notifications.

* When remote publishes a new report, a secure webhook (HMAC-signed) is sent to RW Tool.
* The system verifies signature, creates a new ingestion event if the source is new, and replies with 202 Accepted.
* Duplicate webhooks (same source\_ref) are detected and handled idempotently (no duplicate events).
* Failures (bad signature, malformed data) return errors.
* **Result:** External systems can reliably push updates, ops team can review incoming files, and system avoids duplication.

**UC-SYS2: Notify Subscribers of New Content**

**Goal:** Automatically inform end-users about new reports in their approved domains.

* When a report becomes AVAILABLE after ingestion, system finds all subscribers for that domain.
* Notification service dispatches emails and/or in-app messages to those users.
* If delivery fails, system retries (exponential backoff or similar).
* Every notification is logged for deliverability and audit.
* **Result:** Users get timely alerts about new content, which drives engagement and transparency.

**UC-SYS3: Retention & Archival (Optional)**

**Goal:** Lower storage costs and keep report catalog relevant by archiving old reports.

* Scheduler job moves reports older than N days to ARCHIVED state and migrates files to colder storage (e.g., Glacier).
* Archived reports are hidden from user catalog but remain accessible to Admins/Ops as needed.
* Lifecycle rules and logs ensure data retention policies and compliance.
* **Result:** System manages data volume, respects retention requirements, and maintains discoverability rules.

