1. Executive Summary & Project Rationale

The Auth Signer project is a critical initiative to modernize a fragmented and manual process for managing authorized signers on client accounts. The current state relies on paper-based documents, fragmented storage in systems like **FileNet**, **Chorus**, and **SharePoint**, and manual updates in disparate systems like **WebKYC**. This leads to inaccuracies, delays, and a poor client experience.
The core goal is to establish a single, authoritative, and centralized system—the **Auth Signer Service**—that automates the process from end-to-end. This will create a single source of truth, improve data accuracy, and allow for real-time verification by downstream systems. The project is led by the **Torrent Team**, which is responsible for building the core service, database, and all critical integrations.

2. Core Components & Their Roles

- **Auth Signer Service**: This is the central application built by the Torrent Team. It is a new, independent microservice responsible for:
    - Handling all business logic related to authorized signers (e.g., validation rules, approval flows).
    - Providing secure APIs for both internal and external UIs.
    - Orchestrating all integrations with other bank services.
    - Managing the data in the new Auth Signer Repository.
- **Auth Signer Repository**: A new, dedicated database for the Auth Signer Service. This database will be the **single source of truth**, eliminating data fragmentation. It will store not only the current Auth Signer lists but also a complete, immutable history of all changes for auditing and compliance.
- **External UI**: A new web interface hosted within the **Corporate Connect** portal. Its purpose is to provide a "DIY" self-service experience for the **Client User** persona.
- **Internal UI**: A new web interface for the **Internal Banker** persona (e.g., Client Manager, Product Operations). Its purpose is to give employees a modern, centralized tool for initiating requests and validating client-submitted changes.

3. Detailed Integration Plan: The How, Why, and Where

This is the most critical part of the architecture, detailing every service integration.

3.1 Integration with OCR Service

- **Why**: The OCR integration is the bridge between the old, paper-based world and the new, digital one. It is a key enabler for the MVP, as it allows for the digitization of existing Auth Signer documents without manual transcription.
- **Where**: This integration occurs at the very beginning of the hybrid MVP process, when a banker initiates a request.
- **How**:
    1. An **Internal Banker** uploads a paper document (or a PDF) to the Internal UI.
    2. The Internal UI makes an API call to the **OCR Service**. The payload is the document image/file.

3. The OCR Service processes the image using machine learning models to extract text. It maps this unstructured text to a structured data format (e.g., a JSON object with fields like signerName, authorityLevel, effectiveDate).
4. The OCR Service sends this structured JSON data back to the Auth Signer Service via an API call.

3.2 Integration with Guest User Provisioning Service

- **Why**: This integration is the core of the client's "DIY" experience. It allows a client to securely access the External UI without needing a pre-existing Corporate Connect account, which simplifies the process and avoids a major user provisioning challenge.
- **Where**: This happens after a banker has digitized a document and wants to send it to the client for review.
- **How**:
  1. An **Internal Banker** initiates a "Request Client Attestation" action in the Internal UI.
  2. The Internal UI sends a request to the Auth Signer Service.
  3. The Auth Signer Service makes a secure API call to the **Guest User Provisioning Service**. The payload includes the client's registered email address and the specific record ID they need to access.
  4. The Guest User Provisioning Service generates a unique, single-use URL with an embedded token. It emails this link to the client.
  5. When the client clicks the URL, the External UI validates the token with the Guest User Provisioning Service and grants temporary, scoped access to the specific Auth Signer record.

3.3 Integration with Onboarding Service

- **Why**: This is a strategic integration for long-term data integrity. It ensures that all new Auth Signer data from a client's initial onboarding process is fed directly into the new Auth Signer Repository, guaranteeing clean data from day one.
- **Where**: At the end of the new client onboarding process.
- **How**:
  1. The **Onboarding Service** completes the new client setup.
  2. It makes a final, secure API call to a specific endpoint on the Auth Signer Service (e.g., POST /api/v1/authsigners).
  3. The API payload contains the full, structured Auth Signer list for the new client.
  4. The Auth Signer Service validates and ingests this data, creating the first official record in its repository.

3.4 Integration with WebKYC

- **Why**: WebKYC is the bank’s official system of record for Auth Signer privileges. This integration is crucial for compliance and to ensure that the new system's data is reflected in the official privilege management system. Without this step, privileges are not actually updated.
- **Where**: After a client's request has been fully reviewed and approved by a banker.
- **How**:
  1. An **Internal Banker** clicks "Approve" on a request in the Internal UI.

2. The Internal UI sends a PUT request to the Auth Signer Service to finalize the record.
3. The Auth Signer Service, upon finalizing the record, makes a secure API call to the **WebKYC Service**. The payload includes the client's account ID, the signer's ID, and the action (ADD, REMOVE).
4. WebKYC processes this request and updates the signer's privileges in its system. It returns a confirmation (e.g., 200 OK) to the Auth Signer Service.

3.5 Integration with Downstream Systems (e.g., SWP, STA)

- **Why**: This is the future vision of the project. It will eliminate manual verification by providing other bank systems with real-time access to accurate Auth Signer data.
- **Where**: Whenever a downstream system needs to verify a signer's authority (e.g., before executing a transaction or granting access).
- **How**: The Auth Signer Service will expose a secure, read-only GET API (e.g., /api/v1/authsigners/{accountID}). Downstream systems can call this API to get the latest, most accurate Auth Signer list for an account, without having to maintain their own copies or rely on manual processes.

4. Start-to-Finish Process Walkthrough

Here is a minute-by-minute breakdown of the entire process from a client's perspective to finalization.

1. **Client initiates request**: A client sends an Auth Signer change request (e.g., an email or a paper document) to their banker.
2. **Banker Digitizes Document**: The banker logs into the new **Internal UI**, opens the client's account, and uploads the document. The UI sends this to the **OCR Service**.
3. **Data Extraction & Draft Creation**: The OCR Service extracts the data and sends it back to the **Auth Signer Service**. The service creates a new, draft record in the **Auth Signer Repository**.
4. **Banker sends Guest Link**: The banker, using the Internal UI, generates and sends a guest link to the client. This action triggers the **Guest User Provisioning Service** integration.
5. **Client Review**: The client receives the link, logs in, and views the digitized draft via the **External UI**. They can add, remove, or edit signers.
6. **Client Submits**: The client submits their final changes via the External UI. This updates the record in the **Auth Signer Repository** to a pending state.
7. **Banker Validation & Approval**: The banker is notified of the pending request. They review the changes in the Internal UI, check for any business rule requirements (e.g., two-person approval), and then click "Approve."
8. **Service Finalizes Record**: The "Approve" action triggers the Auth Signer Service to mark the record as finalized in its repository.
9. **WebKYC Update**: **Crucially**, the Auth Signer Service then makes an API call to the **WebKYC Service** to update the official privileges.
10. **Finalization**: WebKYC confirms the update. The process is complete, and the record in the **Auth Signer Repository** is now the official, verified single source of truth for the account.

5. Data Model for Auth Signer Repository

The new repository will store a robust data model for each account.

- accountID: (Primary Key)
- currentSigners: An array of objects, where each object represents a signer and contains:
    - signerID
    - name
    - privilegeLevel
- changeHistory: An array of objects for a full, immutable audit trail, where each object contains:
    - timestamp
    - action (e.g., ADD, REMOVE, EDIT)
    - details (e.g., name of signer added)
    - initiatedBy (e.g., userID of the banker)
- status: (e.g., draft, pending, finalized, rejected)
- lastModified: Timestamp
- lastModifiedBy: userID