Alright, based on all the information from the transcripts, here's a detailed, actionable plan for us as developers to build the Auth Signer system. This breaks down the expectations, phases of work, and the technologies we'll use, serving as a comprehensive guide for our team.

Developer's Guide to the Auth Signer System

Overall Mission: Our primary goal is to build a centralized, automated, and auditable system for managing authorized signers. This isn't just a simple application; it's a foundational service that other systems will depend on. We need to deliver a working MVP for CME by the end of Q4, while building a reusable, long-term solution.

Phase 1: Discovery & Technical Design (Current Focus)

Our immediate priority is to clarify the ambiguous points and lay a solid architectural foundation. This will involve significant collaboration with product, UX, and other teams.

1. Key Discoveries and Assessments (Initial Sprints)

- **Service Request Platform Integration:** We need to investigate how we'll integrate with the Service Request platform (which Susie is working on). We must understand its APIs, data models, and how to trigger our workflows from it. This is a crucial entry point for manual requests.
- **Onboarding Integration:** We must find out how to leverage the data collected during the onboarding process. The goal is to avoid re-entering information. We need to identify the systems that store this data and how we can access it (e.g., APIs, data dumps).
- **WebKYC Reuse:** A key efficiency gain would be to reuse existing WebKYC APIs. We need to get API documentation from Susie and her team and evaluate whether they meet our needs for flagging authorized signers and performing compliance checks. We must confirm if the APIs are robust enough for our purpose or if we'll need to build our own.
- **External User Onboarding:** This is a critical discovery. We need to work with Jessica Roswell and her team to figure out how to onboard an authorized signer to Corporate Connect if they don't already have an account. We need to determine the level of authentication required—is a one-time password (OTP) sufficient, or do we need a full account creation process? We must engage with the security team to get a clear answer here.

2. Technical Design & Prototyping

- **Data Model Design:** We will design the database schema for our central Auth Signer repository. The model must be flexible and robust enough to handle the varying information from different business lines (e.g., authorization types, number of required signers). It needs to be the single source of truth.
- **API Contract Definition:** We will define the API endpoints for our service. This includes:
    - GET endpoints for retrieving ASL lists.
    - POST/PUT endpoints for adding, removing, and updating signers.
    - Endpoints for bulk uploads (we'll need to consider performance and asynchronous processing here).
    - Endpoints for status checks and task management.

- **Authentication & Entitlements:** The API design must incorporate an entitlement check on every call to ensure the user (client or internal) has the right to access or modify the ASL for a given account.

Phase 2: Core MVP Development (Q4 2025)

Once the discovery and design are complete, we will begin building the core functionality. Our immediate goal is a working MVP for CME. We must build this service with reusability in mind so that it can easily be adapted for IT&C and other business lines later.

1. Backend Development (Spring Boot)

- **Central Database & CRUD Operations:** We will build the core Spring Boot service to interact with the SQL Server database. This includes all the basic CRUD (Create, Read, Update, Delete) operations for managing authorized signer records.
- **Validation Engine:** We'll implement the business logic for validating change requests. This includes single-signer validation and the more complex multi-person approval workflows.
- **WebKYC Integration:** We will write the code to call the WebKYC APIs to perform automated compliance checks for new signers.
- **System-of-Record Integrations:** We will build the API integrations to update other systems of record, specifically FileNet for CME. We must determine if we'll use a "push" or "pull" model and who is responsible for initiating the data flow.
- **Digital Signature Support:** We will build the technical framework for digital signatures, working closely with the security and legal teams to ensure compliance. We'll need to understand if we can use an internal service or a third-party vendor.

2. Frontend Development (React)

- **Dashboard & View:** We will develop the Auth Signer section within the Corporate Connect portal. This includes the dashboard showing the current ASL and the status of requests.
- **Maintenance Forms:** We'll build the forms for adding, removing, and attesting to signers, ensuring the UI is intuitive and easy for the client to use.
- **Document Generation:** We'll implement the logic to generate exportable documents (e.g., PDFs) of the ASL list, ensuring they include business-line-specific verbiage as required.
- **Task Management UI:** For internal users, we'll build a UI that displays their assigned tasks for multi-person approvals and allows them to approve or reject requests.

Phase 3: Long-Term Roadmap & Expansion (2026+)

This is what we'll work towards after the initial MVP.

- **Onboarding Integration:** We'll fully integrate our service with the ongoing onboarding initiatives for GSF and GCT, ensuring ASL data is fed directly into our service from the beginning.
- **AI/OCR Implementation:** We will partner with the Cognizant team to ingest historical ASL documents. We'll build the necessary APIs to receive their processed data and populate

our database, solving the data migration problem.
- **Bulk Upload Functionality:** We will finalize the implementation for bulk uploads, likely using **Kafka** to handle the high volume and ensure reliability.
- **Advanced Features:** We can add features like proactive notifications for attestation, advanced entitlement management, and direct integration with transaction systems (e.g., a loans or wire transfer system) to automatically check ASL data before releasing funds.

Our Tooling and Systems

- **Backend: Spring Boot** (Java), **SQL Server**.
- **Frontend: React** (JavaScript/TypeScript), integrated into **Corporate Connect**.
- **Code Management:** We'll use our standard Git repository and CI/CD pipelines.
- **Message Queue (Potential): Kafka** for handling high-volume, asynchronous tasks like bulk uploads.
- **APIs:** We will use a RESTful API design.
- **Integration Points: Service Request platform**, **WebKYC**, **FileNet**, **Cognizant's AI service**, and a centralized **task management system**.

Our immediate next steps are to hold follow-up meetings with the relevant teams to get the answers we need. We can't start coding the core logic until we have clarity on these key dependencies. The plan is to get the design and discovery completed so that we can hit the ground running once our new team members are onboard.