KYC Service Architecture & Flow

Overview

The KYC Service is a microservice that handles identity verification for user authentication and compliance. It supports two KYC providers: **Plaid** and **Sumsub**, with the ability to switch between them using environment configuration.

System Architecture

```
graph TB
    subgraph "Frontend Application"
        UI[User Interface]
        AuthUI[Auth Pages]
    end

    subgraph "KYC Microservice Port 3002"
        Router[Express Routes]
        Auth[Auth Middleware]
        Controller[KYC Controller]
        PlaidSvc[Plaid Service]
        SumsubSvc[Sumsub Service]
        StorageSvc[Storage Service]
    end

    subgraph "External Providers"
        PlaidAPI[Plaid API]
        SumsubAPI[Sumsub API]
    end

    subgraph "Shared Database PostgreSQL"
        UsersTable[(users)]
        SessionTable[(session)]
        KYCTable[(kyc_verification)]
        MembershipTable[(user_memberships)]
        KYCStatusTable[(kyc_statuses)]
        AddressTable[(user_addresses)]
    end

    subgraph "Storage"
        Supabase[Supabase Storage]
    end

    UI --> Router
    AuthUI --> Router
    Router --> Auth
    Auth --> SessionTable
    Auth --> UsersTable
    Auth --> Controller
```

```
    Controller --> PlaidSvc
    Controller --> SumsubSvc
    Controller --> StorageSvc
    Controller --> KYCTable
    Controller --> MembershipTable
    Controller --> UsersTable
    Controller --> AddressTable

    PlaidSvc --> PlaidAPI
    SumsubSvc --> SumsubAPI
    StorageSvc --> Supabase

    SumsubAPI -.Webhook.-> Router

    KYCTable --> KYCStatusTable
```

Core Components

1. **Authentication Middleware (auth.middleware.ts)**

Validates user sessions before allowing KYC operations:

- Checks for Better Auth session cookie
- Falls back to Authorization Bearer token
- Queries session from shared database
- Validates session expiration
- Attaches user info to request

2. **KYC Controller (kyc.controller.ts)**

Main business logic handler with support for both providers:

- Creates link tokens or access tokens
- Manages applicant creation (Sumsub)
- Handles document uploads
- Triggers verification process
- Processes verification results
- Updates database records
- Manages webhooks (Sumsub)

3. **Provider Services**

Plaid Service (plaid.service.ts)

- Creates Identity Verification link tokens
- Retrieves verification status
- Extracts risk scores
- Maps Plaid status to internal status

Sumsub Service (sumsub.service.ts)

- Generates HMAC signatures for API auth
- Creates applicants
- Uploads documents (front/back)

- Generates SDK tokens for liveness
- Starts verification process
- Retrieves applicant data
- Maps Sumsub status to internal status
- Extracts risk scores and document info

4. **Storage Service** (storage.service.ts)

- Uploads documents to Supabase Storage
- Maintains backup copies of verification documents
- Organized by `userId/applicantId/type_timestamp.ext`

KYC Verification Flows

Flow 1: Plaid Identity Verification

```
sequenceDiagram
    participant User
    participant Frontend
    participant KYC Service
    participant Plaid API
    participant Database

    User->>Frontend: Initiates KYC
    Frontend->>KYC Service: POST /kyc/create-link-token
    KYC Service->>Plaid API: Create IDV Link Token
    Plaid API-->>KYC Service: Return Link Token
    KYC Service-->>Frontend: Return Link Token

    Frontend->>User: Display Plaid Link UI
    User->>Plaid API: Complete Verification
    Plaid API-->>Frontend: Return Session ID

    Frontend->>KYC Service: POST /kyc/verify {sessionId}
    KYC Service->>Plaid API: Get Verification Result
    Plaid API-->>KYC Service: Verification Data
    KYC Service->>Database: Store Verification Record
    KYC Service->>Database: Update user_memberships status
    KYC Service-->>Frontend: Return Status
    Frontend-->>User: Show Verification Result
```

Flow 2: Sumsub Direct API Integration

```
sequenceDiagram
    participant User
    participant Frontend
    participant KYC Service
    participant Sumsub API
    participant Storage
    participant Database
```

```
User->>Frontend: Initiates KYC
Frontend->>KYC Service: POST /kyc/create-applicant
KYC Service->>Database: Fetch user details
Database-->>KYC Service: User info
KYC Service->>Sumsub API: Create Applicant
Sumsub API-->>KYC Service: Applicant ID
KYC Service->>Database: Store applicant ID (PENDING)
KYC Service-->>Frontend: Return Applicant ID

User->>Frontend: Upload ID Documents
Frontend->>KYC Service: POST /kyc/upload-document {front, back}
KYC Service->>Storage: Store documents
Storage-->>KYC Service: Storage paths
KYC Service->>Database: Update document URLs
KYC Service->>Sumsub API: Upload front document
Sumsub API-->>KYC Service: Upload success
KYC Service->>Sumsub API: Upload back document
Sumsub API-->>KYC Service: Upload success
KYC Service-->>Frontend: Documents uploaded

User->>Frontend: Complete Liveness Check
Frontend->>KYC Service: POST /kyc/get-sdk-token
KYC Service->>Sumsub API: Get SDK Token
Sumsub API-->>KYC Service: SDK Token
KYC Service-->>Frontend: Return SDK Token
Frontend->>Sumsub API: Perform Liveness (SDK)
Sumsub API-->>Frontend: Liveness complete

Frontend->>KYC Service: POST /kyc/start-verification
KYC Service->>Sumsub API: Request to start review
Sumsub API-->>KYC Service: Verification started
KYC Service-->>Frontend: Return status (IN_REVIEW)

Note over Sumsub API: Manual/Auto Review

Sumsub API->>KYC Service: POST /kyc/webhook/applicant-reviewed
KYC Service->>Sumsub API: Fetch full applicant data
Sumsub API-->>KYC Service: Complete applicant info
KYC Service->>Database: Update kyc_verification (VERIFIED/REJECTED)
KYC Service->>Database: Update user_memberships status
KYC Service-->>Sumsub API: 200 OK

User->>Frontend: Check status
Frontend->>KYC Service: GET /kyc/check-status
KYC Service->>Database: Query kyc_verification
Database-->>KYC Service: Current status
KYC Service-->>Frontend: Return VERIFIED/REJECTED
Frontend-->>User: Display result
```

Flow 3: Sumsub Hosted Verification (QR Code)

```
sequenceDiagram
    participant User
    participant Admin
    participant KYC Service
    participant Sumsub API
    participant Database

    Admin->>KYC Service: POST /kyc/generate-hosted-link {userId}
    KYC Service->>Sumsub API: Generate Access Token
    Sumsub API-->>KYC Service: Access Token
    KYC Service->>Sumsub API: Generate Hosted URL
    Sumsub API-->>KYC Service: Hosted URL
    KYC Service-->>Admin: Return QR Code URL

    Admin->>User: Display QR Code
    User->>Sumsub API: Scan QR & Complete Verification

    Note over Sumsub API: Review Process

    Sumsub API->>KYC Service: POST /kyc/webhook/applicant-reviewed
    KYC Service->>Database: Update verification status
    KYC Service-->>Sumsub API: 200 OK
```

Database Schema

kyc_verification Table

Key fields storing verification data:

Field	Type	Description
`id`	UUID	Primary key
`user_id`	UUID	Foreign key to users
`kyc_status_id`	Integer	Status reference (1=Pending, 2=In Review, 3=Verified, 4=Rejected, 5=Expired)
`verification_provider`	Text	'plaid' or 'sumsub'
`provider_verification_id`	Text	Session ID (Plaid) or Applicant ID (Sumsub)
`verification_method`	Text	Verification method used
`document_type`	Text	ID type (passport, driver_license, etc.)
`document_front_url`	Text	Storage path for front image
`document_back_url`	Text	Storage path for back image
`selfie_url`	Text	Storage path for selfie
`verification_data`	JSONB	Full provider response

`risk_score`	Integer	0-100 risk score
`verified_at`	Timestamp	When verification completed
`rejection_reason`	Text	Reason if rejected

kyc_statuses Reference Table

ID	Name	Description
1	Pending	Initial state
2	In Review	Submitted for review
3	Verified	Successfully verified
4	Rejected	Verification failed
5	Expired	Verification expired

API Endpoints

User Endpoints (Require Auth)

Method	Endpoint	Description
`POST`	`/kyc/create-link-token`	Create Plaid link token or Sumsub access token
`POST`	`/kyc/create-applicant`	Create Sumsub applicant
`POST`	`/kyc/upload-document`	Upload ID documents
`POST`	`/kyc/get-sdk-token`	Get Sumsub SDK token for liveness
`POST`	`/kyc/start-verification`	Start verification review
`POST`	`/kyc/verify`	Submit verification results (Plaid)
`GET`	`/kyc/check-status`	Get current KYC status
`POST`	`/kyc/generate-hosted-link`	Generate hosted verification URL

Admin Endpoints

Method	Endpoint	Description
`GET`	`/kyc/admin/verified-users`	List all verified users
`GET`	`/kyc/admin/stats`	Get KYC statistics
`GET`	`/kyc/admin/applicant/:id`	Get applicant data (Sumsub)
`GET`	`/kyc/admin/applicant/:id/status`	Get applicant status (Sumsub)

Webhook Endpoint (No Auth)

Method	Endpoint	Description
POST	/kyc/webhook/applicant-reviewed	Sumsub webhook for status updates

---

Status Mapping

Sumsub → Internal Status

Sumsub Status	Internal Status	ID
init, pending	Pending	1
queued, onHold	In Review	2
completed + GREEN	Verified	3
completed + RED	Rejected	4
Other	Pending	1

Plaid → Internal Status

Plaid Status	Internal Status	ID
success	Verified	3
failed	Rejected	4
expired	Expired	5
pending_review	In Review	2
requires_input, active	Pending	1

Risk Scoring

Sumsub Risk Score

Calculated from `reviewResult.reviewAnswer`:

- `GREEN`: 0 (lowest risk)
- `YELLOW`: 50 (medium risk)
- `RED`: 100 (highest risk)

Plaid Risk Score

Calculated from verification steps:

- Each failed step: +25
- Each manually approved step: +10
- Capped at 100

Configuration

Environment Variables

Variable	Required	Description
`KYC_PROVIDER`	Yes	'plaid' or 'sumsub'
`PLAID_CLIENT_ID`	If Plaid	Plaid client ID
`PLAID_SECRET`	If Plaid	Plaid secret key
`PLAID_ENV`	If Plaid	'sandbox' or 'production'
`PLAID_IDV_TEMPLATE_ID`	If Plaid	Template ID for IDV
`SUMSUB_APP_TOKEN`	If Sumsub	Sumsub app token
`SUMSUB_SECRET_KEY`	If Sumsub	Sumsub secret key
`SUMSUB_BASE_URL`	If Sumsub	API base URL
`SUMSUB_LEVEL_NAME`	If Sumsub	Verification level
`DATABASE_URL`	Yes	PostgreSQL connection
`SUPABASE_URL`	Yes	Supabase project URL
`SUPABASE_KEY`	Yes	Supabase service key

Key Features

✅ Dual Provider Support

- Environment-based switching between Plaid and Sumsub
- Unified API regardless of provider
- Provider-specific features accessible when needed

✅ Document Storage

- Documents uploaded to Supabase Storage
- Organized directory structure
- Maintains verification audit trail

✅ Webhook Integration

- Real-time status updates from Sumsub
- Automatic database synchronization
- Updates user membership status

✅ Session Validation

- Shared Better Auth session
- Cookie and Bearer token support
- Secure cross-service authentication

✅ Status Synchronization

When KYC status changes to VERIFIED:

1. Updates `kyc_verification.kyc_status_id = 3`
2. Updates `kyc_verification.verified_at` timestamp
3. Updates `user_memberships.verification_status_id = 2` (Verified)

When REJECTED:

- Updates `kyc_verification.kyc_status_id = 4`
- Updates `user_memberships.verification_status_id = 3` (Failed)

Security Measures

- ✅ Session-based authentication via Better Auth
- ✅ HMAC signature authentication for Sumsub API
- ✅ Secure webhook validation
- ✅ Document encryption in storage
- ✅ Audit logging for all operations
- ✅ Risk score calculation for fraud detection
- ✅ IP address and user agent tracking

Integration Points

1. Main Application

Proxies requests from frontend to KYC service while maintaining session cookies.

2. User Management

Updates `user_memberships` table based on verification results.

3. Supabase Storage

Stores verification documents with proper access control.

4. External Providers

- **Plaid**: IDV flow with Link UI
- **Sumsub**: Direct API or hosted verification

Monitoring & Audit

Admin Capabilities

- View all verified users
- Access applicant data for compliance
- Generate KYC statistics dashboard
- Review verification history

Audit Trail

- All verification attempts logged
- Document upload tracking
- Status change history
- Provider responses stored in JSONB

Error Handling

Provider Failures

- Graceful degradation when provider unavailable
- Detailed error messages logged
- User-friendly error responses

Validation Errors

- Document format validation
- Required field validation
- Session expiration handling

Webhook Failures

- Idempotent webhook processing
- Detailed logging for debugging
- Returns 200 OK even if record not found

Future Enhancements

> *[!NOTE] Potential improvements for the service*

- Multi-document upload support
- Real-time verification status polling
- Enhanced fraud detection algorithms
- Integration with additional KYC providers
- Automated document quality checks
- OCR for document data extraction