MediNote Application Programming Interface (API) Comprehensive Documentation Suite

Executive Summary & Architectural Overview

This comprehensive technical documentation suite provides exhaustive API endpoint specifications for the MediNote mobile application ecosystem, encompassing both client-side and server-side architectural components within a distributed microservices architecture paradigm.

Infrastructure & Deployment Topology

The application leverages a multi-tiered infrastructure deployment strategy with the following endpoint resolution matrices:

Authentication & Authorization Framework

The API implements a stateless authentication mechanism utilizing JWT (JSON Web Token) bearer token authentication protocol. All service endpoints mandate the inclusion of the authorization header with the following standardized format:

```
None
Authorization: Bearer <encoded_jwt_token>
```

Security Considerations:

- Token expiration policies are enforced at the service layer
- Automatic token refresh mechanisms are implemented client-side
- 401 Unauthorized responses trigger automatic authentication flow reinitialization

Patient Entity Management & Data Persistence Layer

Patient Collection Retrieval Endpoint

HTTP Method: `GET`
Endpoint Path: `/v1/patients`
Service Category: Patient Management Service Layer

This endpoint facilitates the retrieval of patient entity collections associated with a specific user context within the application's data persistence layer. The endpoint implements pagination-agnostic retrieval mechanisms and returns a structured collection of patient entities.

Request Specification

Query Parameter Matrix:

- `userId` (string, required, immutable): Unique identifier for the user entity within the database schema, utilized for data isolation and access control enforcement

Response Schema Definition

The endpoint returns a JSON-encoded response object containing a collection of patient entities:

```json
JSON
{
  "patients": [
    {
      "id": "patient_123",
      "name": "John Doe"
    }
  ]
}
```

Schema Validation Rules:

- `patients`: Array of patient entity objects
- `id`: String identifier conforming to UUID v4 specification
- `name`: Human-readable patient identifier string

User Entity Resolution & Database ID Mapping Endpoint

HTTP Method: `GET`
Endpoint Path: `/users/asd3fd2faec`
Service Category: User Identity Resolution Service

This specialized endpoint implements a reverse lookup mechanism for resolving user entity database identifiers through email address resolution. The endpoint operates within the backend service layer and provides essential identity mapping functionality for cross-service communication.

Request Specification

Query Parameter Matrix:

- `email` (string, required, validated): Canonical email address string conforming to RFC 5322 specification, utilized as the primary lookup key for user entity resolution

Response Schema Definition

The endpoint returns a minimal response object containing the resolved user entity identifier:

```json
{
  "id": "user_123"
}
```

Response Validation:

- `id`: String identifier representing the resolved user entity's primary key within the database schema

Patient Entity Creation & Persistence Endpoint

HTTP Method: `POST`
Endpoint Path: `/v1/add-patient-ext`
Service Category: Patient Management Service Layer

This endpoint implements the patient entity creation workflow within the application's data persistence layer. The endpoint enforces business logic validation rules and establishes the necessary entity relationships within the database schema.

Request Specification

Request Body Schema:

```json
{
  "name": "John Doe",
  "userId": "user_123"
}
```

Field Validation Rules:

- `name`: String value representing the patient's canonical identifier, subject to length and character validation constraints
- `userId`: String identifier referencing the associated user entity within the database schema

Response Schema Definition

The endpoint returns a comprehensive response object containing the newly created patient entity:

```json
{
  "patient": {
    "id": "patient_123",
    "name": "John Doe",
    "user_id": "user_123",
    "pronouns": null
  }
}
```

Response Field Specifications:

- `patient`: Object containing the complete patient entity representation
- `id`: System-generated unique identifier for the patient entity
- `name`: Canonical patient identifier as provided in the request
- `user_id`: Foreign key reference to the associated user entity
- `pronouns`: Optional field for patient pronoun preferences (nullable)

Patient Entity Detail Retrieval & Comprehensive Data Access Endpoint

HTTP Method: `GET`
Endpoint Path: `/v1/patient-details/{patientId}`
Service Category: Patient Management Service Layer

This endpoint implements comprehensive patient entity data retrieval functionality, providing access to the complete patient profile including demographic information, medical history, and contextual metadata. The endpoint enforces strict access control policies and implements data sanitization mechanisms.

Request Specification

Path Parameter Matrix:

- `patientId` (string, required, immutable): Unique identifier for the target patient entity within the database schema, utilized for entity resolution and access control validation

Response Schema Definition

The endpoint returns a comprehensive patient entity object containing all available patient data:

```json
{
  "id": "patient_123",
  "name": "John Doe",
  "pronouns": "he/him",
  "email": "john@example.com",
  "background": "Patient background information",
  "medical_history": "Previous medical conditions",
  "family_history": "Family medical history",
  "social_history": "Social history information",
  "previous_treatment": "Previous treatments"
}
```

Response Field Specifications:

- `id`: Primary key identifier for the patient entity
- `name`: Canonical patient identifier string
- `pronouns`: Patient pronoun preferences for respectful communication
- `email`: Contact email address for the patient entity
- `background`: Contextual background information about the patient
- `medical_history`: Historical medical condition documentation
- `family_history`: Familial medical history information
- `social_history`: Social context and lifestyle information
- `previous_treatment`: Documentation of prior medical interventions

Get Patient Sessions

GET `/v1/fetch-session-by-patient/{patientId}`

Get all sessions for a specific patient.

Path Parameters:

- `patientId` (string, required): Patient ID

Response:

```JSON
{
  "sessions": [
    {
      "id": "session_123",
      "date": "2024-01-15",
      "session_title": "Initial Consultation",
      "session_summary": "Patient consultation summary",
      "start_time": "2024-01-15T10:00:00Z"
    }
  ]
}
```

Get All Sessions

GET `/v1/all-session`

Get all sessions for a user with patient details.

Query Parameters:

- `userId` (string, required): User database ID

Response:

```JSON
{
  "sessions": [
    {
      "id": "session_123",
      "user_id": "user_123",
      "patient_id": "patient_123",
      "session_title": "Initial Consultation",
      "session_summary": "Patient consultation summary",
      "transcript_status": "completed",
      "transcript": "Full transcript text...",
      "status": "completed",
      "date": "2024-01-15",
      "start_time": "2024-01-15T10:00:00Z",
      "end_time": "2024-01-15T10:30:00Z",
      "patient_name": "John Doe",
      "pronouns": "he/him",
      "email": "john@example.com",
      "background": "Patient background",
      "duration": "30 minutes",
      "medical_history": "Previous conditions",
      "family_history": "Family history",
      "social_history": "Social history",
      "previous_treatment": "Previous treatments",
      "patient_pronouns": "he/him",
      "clinical_notes": []
    }
  ],
  "patientMap": {
    "patient_123": {
      "name": "John Doe",
      "pronouns": "he/him"
    }
  }
}
```

Template Management & Configuration Service Layer

User Template Collection Retrieval & Configuration Management Endpoint

HTTP Method: `GET`
Endpoint Path: `/v1/fetch-default-template-ext`
Service Category: Template Management Service Layer

This endpoint implements the template configuration retrieval mechanism, providing access to user-specific template collections within the application's configuration management system. The endpoint supports multiple template types and implements hierarchical template resolution strategies.

Request Specification

Query Parameter Matrix:

- `userId` (string, required, immutable): Unique identifier for the user entity within the database schema, utilized for template collection filtering and access control enforcement

Response Schema Definition

The endpoint returns a structured response object containing the user's available template collection:

```json
{
  "success": true,
  "data": [
    {
      "id": "template_123",
      "title": "New Patient Visit",
      "type": "default"
    },
    {
      "id": "template_456",
      "title": "Follow-up Visit",
      "type": "predefined"
    }
  ]
}
```

Response Field Specifications:

- `success`: Boolean indicator of operation completion status
- `data`: Array of template entity objects available to the user
- `id`: Unique identifier for the template entity
- `title`: Human-readable template identifier
- `type`: Template classification type (default, predefined, custom)

Recording Session Management & Audio Processing Service Layer

Recording Session Initialization & Entity Creation Endpoint

HTTP Method: `POST`
Endpoint Path: `/v1/upload-session`
Service Category: Recording Management Service Layer

This endpoint implements the recording session initialization workflow, establishing a new session entity within the application's data persistence layer and configuring the necessary infrastructure for audio data processing and storage. The endpoint enforces session state management policies and implements resource allocation mechanisms.

Request Specification

Request Body Schema:

```json
{
  "patientId": "patient_123",
  "userId": "user_123",
  "patientName": "John Doe",
  "status": "recording",
  "startTime": "2024-01-15T10:00:00Z",
  "templateId": "new_patient_visit"
}
```

Field Validation Rules:

- `patientId`: String identifier referencing the associated patient entity
- `userId`: String identifier referencing the user entity initiating the session
- `patientName`: Canonical patient identifier for display purposes
- `status`: Session state indicator (recording, completed, failed)
- `startTime`: ISO 8601 formatted timestamp indicating session initiation
- `templateId`: Template identifier for session configuration

Response Schema Definition

The endpoint returns a minimal response object containing the newly created session identifier:

```json
{
  "id": "session_123"
}
```

Response Field Specifications:

- `id`: System-generated unique identifier for the recording session entity

Presigned URL Generation & Cloud Storage Access Token Provisioning Endpoint

HTTP Method: `POST`
Endpoint Path: `/v1/get-presigned-url`
Service Category: Cloud Storage Integration Service Layer

This endpoint implements the presigned URL generation mechanism for secure, time-limited access to Google Cloud Storage resources. The endpoint facilitates the upload of audio chunk data to cloud storage infrastructure while maintaining security through cryptographic signature validation and temporal access restrictions.

Request Specification

Request Body Schema:

```json
{
  "sessionId": "session_123",
  "chunkNumber": 1,
  "mimeType": "audio/wav"
}
```

Field Validation Rules:

- `sessionId`: String identifier referencing the active recording session entity
- `chunkNumber`: Integer value indicating the sequential position of the audio chunk within the session
- `mimeType`: MIME type specification for the audio data format (audio/wav, audio/mp3, etc.)

Response Schema Definition

The endpoint returns a comprehensive response object containing cloud storage access information:

```json
{
  "url": "https://storage.googleapis.com/bucket/path/to/file",
  "gcsPath": "sessions/session_123/chunk_1.wav",
  "publicUrl": "https://storage.googleapis.com/bucket/public/path/to/file"
}
```

Response Field Specifications:

- `url`: Presigned URL for direct upload to Google Cloud Storage with embedded authentication credentials
- `gcsPath`: Canonical path within the Google Cloud Storage bucket for the uploaded file
- `publicUrl`: Publicly accessible URL for the uploaded file (if applicable)

Audio Chunk Upload Notification & Processing Pipeline Trigger Endpoint

HTTP Method: `POST`
Endpoint Path: `/v1/notify-chunk-uploaded`
Service Category: Audio Processing Pipeline Service Layer

This endpoint implements the chunk upload notification mechanism, triggering the audio processing pipeline upon successful upload of audio data chunks to cloud storage. The endpoint facilitates asynchronous processing workflows and implements event-driven architecture patterns for audio data processing and transcription services.

Request Specification

Request Body Schema:

```json
{
  "sessionId": "session_123",
  "gcsPath": "sessions/session_123/chunk_1.wav",
  "chunkNumber": 1,
  "isLast": false,
  "totalChunksClient": 5,
  "publicUrl": "https://storage.googleapis.com/bucket/public/path/to/file",
  "mimeType": "audio/wav",
  "selectedTemplate": "New Patient Visit",
  "selectedTemplateId": "new_patient_visit",
  "model": "fast"
}
```

Field Validation Rules:

- `sessionId`: String identifier referencing the active recording session entity
- `gcsPath`: Canonical path within Google Cloud Storage for the uploaded chunk
- `chunkNumber`: Integer value indicating the sequential position of the chunk
- `isLast`: Boolean flag indicating whether this is the final chunk in the sequence
- `totalChunksClient`: Integer value representing the total number of chunks expected
- `publicUrl`: Publicly accessible URL for the uploaded chunk
- `mimeType`: MIME type specification for the audio data format
- `selectedTemplate`: Human-readable template identifier
- `selectedTemplateId`: Canonical template identifier for processing configuration
- `model`: Processing model specification for transcription services

Response Schema Definition

The endpoint returns an empty response object indicating successful notification processing:

```
JSON
{}
```

Response Specifications:

- Empty response body indicates successful notification processing
- HTTP 200 status code confirms successful endpoint execution

Direct Cloud Storage Integration & Binary Data Transfer Layer

Binary Audio Data Transfer & Cloud Storage Direct Upload Endpoint

HTTP Method: `PUT`
Endpoint Path: `{presignedUrl}` (Dynamic URL Resolution)
Service Category: Cloud Storage Direct Integration Layer

This endpoint implements direct binary data transfer to Google Cloud Storage infrastructure utilizing presigned URL authentication mechanisms. The endpoint operates outside the traditional service layer architecture and implements direct cloud storage integration patterns for optimal performance and reduced latency in audio data transfer operations.

Request Specification

Dynamic URL Resolution: The endpoint URL is dynamically generated through the presigned URL generation service and contains embedded authentication credentials and temporal access restrictions.

Required Headers:

- `Content-Type`: `audio/wav` (MIME type specification for audio data format)
- `Authorization`: `Bearer <token>` (Authentication token for request validation)

Request Body: Raw binary audio data encoded in WAV format, transmitted as a binary stream without additional encoding or transformation.

Response Schema Definition

The endpoint returns an empty response body upon successful upload completion:

Response Specifications:

- Empty response body indicates successful binary data transfer
- HTTP 200 status code confirms successful upload operation
- No additional metadata or confirmation data is returned

Error Handling & Exception Management Framework

HTTP Status Code Matrix & Error Response Specifications

The API implements a comprehensive error handling framework utilizing standardized HTTP status codes and structured error response schemas. All endpoints return appropriate status codes based on operation outcomes and error conditions.

Standard HTTP Status Code Implementation

- `200` - **Success**: Operation completed successfully with valid response data
- `201` - **Created**: Resource entity created successfully within the data persistence layer
- `400` - **Bad Request**: Client request validation failure or malformed request syntax
- `401` - **Unauthorized**: Authentication token validation failure or missing authorization credentials
- `404` - **Not Found**: Requested resource entity does not exist within the system
- `500` - **Internal Server Error**: Unhandled server-side exception or system failure

Error Response Schema Definition

Error responses implement a standardized structure for consistent error handling across all service endpoints:

```json
{
  "error": "Error message",
  "details": "Additional error details"
}
```

Error Response Field Specifications:

- `error`: Primary error message describing the failure condition
- `details`: Additional contextual information for error diagnosis and resolution

Implementation Notes & Architectural Considerations

Authentication & Security Implementation

1. **Token-Based Authentication**: All service endpoints mandate valid Bearer token authentication for request authorization
2. **Content-Type Validation**: POST requests require `Content-Type: application/json` header specification for proper request processing
3. **Multi-Service Architecture**: The application implements different base URLs for distinct service layers (main API vs backend API)

Data Transfer & Storage Architecture

4. **Cloud Storage Integration**: Audio chunk data is transferred directly to Google Cloud Storage utilizing presigned URL authentication mechanisms
5. **Chunked Processing**: Audio recording data is processed in discrete chunks to optimize performance and ensure system reliability
6. **Template System**: The application supports hierarchical template classification (default, predefined, custom) for session configuration

Service Layer Architecture Reference

This comprehensive documentation suite is derived from the following service layer implementations:

Core Service Files

- `src/services/patient.service.ts` - Patient entity management and data persistence service layer
- `src/services/template.service.ts` - Template configuration and management service layer
- `src/services/recording.service.ts` - Recording session management and audio processing service layer
- `src/services/utils.ts` - Utility functions and helper service implementations (currently in maintenance mode)

Application Layer Integration

- `src/app/(active-session)/page.tsx` - Direct Google Cloud Storage integration and binary data transfer implementation