**API Requirements Document**

**Overview**

This document outlines the API requirements for the Airiam RAG Service frontend application. It details the endpoints, request/response formats, authentication mechanisms, and error handling needed to support the various dashboard functionalities. This guide serves as a reference for backend developers to implement the necessary API endpoints that the frontend will consume.

The API follows RESTful design principles, using standard HTTP methods and status codes. All endpoints require proper authentication through JWT tokens (or other specified auth mechanisms) and return responses in JSON format. The API should implement proper error handling, validation, and security measures.

**Authentication and Authorization**

**Authentication Endpoints**

**User Authentication**

* **Login**: POST /auth/login
  + Request: { email: string, password: string }
  + Response: { user: User, token: string }
  + Used by: Login page
* **Register**: POST /auth/register
  + Request: { name: string, email: string, password: string, tenantId?: string }
  + Response: { user: User, token: string }
  + Used by: Registration page
* **Current User**: GET /auth/me
  + Response: User object
  + Used by: All authenticated pages to verify current user
* **Logout**: POST /auth/logout
  + Request: Empty or refresh token
  + Response: Success message
  + Used by: Header component for logout function

**Authorization Requirements**

* **JWT Token**: Include in Authorization header (Bearer <token>)
* **Role-Based Access Control**: Endpoints check user's role for permission
* **Tenant Isolation**: Multi-tenant endpoints enforce tenant access boundaries

**File Management API**

**File Operations**

**File Retrieval**

* **List Files**: GET /files
  + Query params: { search?: string, tags?: string[], categories?: string[], page?: number, limit?: number }
  + Response: { files: File[], total: number, page: number, limit: number }
  + Used by: File Analysis dashboard
* **Get File**: GET /files/{id}
  + Response: File object with all details
  + Used by: File detail view

**File Upload and Management**

* **Upload Files**: POST /files/upload
  + Request: Multipart form data with files and options
  + Response: { files: File[] } (newly created files)
  + Used by: Add Files modal
* **Add Cloud Files**: POST /files/cloud
  + Request: { provider: string, files: string[], storeFullContent?: boolean, suggestTags?: boolean }
  + Response: { files: File[] } (newly added files)
  + Used by: Cloud connector in Add Files modal
* **Delete File**: DELETE /files/{id}
  + Response: Success message
  + Used by: File deletion functionality

**File Metadata Management**

* **Update Tags**: PUT /files/{id}/tags
  + Request: { tags: FileTag[] }
  + Response: Updated File object
  + Used by: Tag management modal
* **Update Categories**: PUT /files/{id}/categories
  + Request: { categories: FileCategory[] }
  + Response: Updated File object
  + Used by: Category management functionality

**Tag and Category Management**

* **List Tags**: GET /tags
  + Response: FileTag[]
  + Used by: Tag selection dropdowns and filters
* **List Categories**: GET /categories
  + Response: FileCategory[]
  + Used by: Category selection dropdowns and filters

**Chat and Conversation API**

**Conversation Management**

* **List Conversations**: GET /conversations
  + Query params: Optional filtering
  + Response: Conversation[] (list of conversation summaries)
  + Used by: Chat dashboard
* **Get Conversation**: `

Retry

Claude hit the max length for a message and has paused its response. You can write Continue to keep the chat going.

GE

continue

Edit

**API Requirements Document (continued)**

**Conversation Management (continued)**

* **Get Conversation**: GET /conversations/{id}
  + Response: Conversation (full conversation with messages)
  + Used by: Individual chat conversation page
* **Create Conversation**: POST /conversations
  + Request: { title: string, fileContext?: string[] }
  + Response: New Conversation object
  + Used by: Create conversation modal
* **Delete Conversation**: DELETE /conversations/{id}
  + Response: Success message
  + Used by: Conversation deletion functionality
* **Update Context**: PUT /conversations/{id}/context
  + Request: { fileContext: string[] }
  + Response: Updated Conversation object
  + Used by: File context selector in chat interface

**Message Management**

* **Send Message**: POST /conversations/{id}/messages
  + Request: { content: string, fileContext?: string[] }
  + Response: Message object (AI's response message)
  + Used by: Message input in conversation view

**Analytics API**

**Dashboard Analytics**

* **Usage Statistics**: GET /analytics/usage-stats
  + Query params: { range: 'day' | 'week' | 'month' | 'year' }
  + Response: { totalQueries: number, totalFiles: number, totalTokensConsumed: number, averageResponseTime: number }
  + Used by: Dashboard statistics cards
* **Activity Log**: GET /analytics/activity-log
  + Query params: { range: 'day' | 'week' | 'month' | 'year' }
  + Response: Array of activity log entries
  + Used by: Recent activity widget
* **Credit Consumption**: GET /analytics/credit-consumption
  + Query params: { range: 'day' | 'week' | 'month' | 'year' }
  + Response: Array of { date: string, credits: number } entries
  + Used by: Credit consumption chart
* **Query Distribution**: GET /analytics/query-distribution
  + Query params: { range: 'day' | 'week' | 'month' | 'year' }
  + Response: Array of { category: string, count: number } entries
  + Used by: Query distribution chart
* **Top Documents**: GET /analytics/top-documents
  + Query params: { range: 'day' | 'week' | 'month' | 'year' }
  + Response: Array of { id: string, name: string, type: string, accessCount: number } entries
  + Used by: Top documents table

**User Management API**

**User Operations**

* **List Users**: GET /users
  + Query params: Optional filtering and pagination
  + Response: { users: User[], total: number, page: number, limit: number }
  + Used by: User management dashboard
* **Get User**: GET /users/{id}
  + Response: User object with all details
  + Used by: User detail panel
* **Create User**: POST /users
  + Request: { name: string, email: string, role: string, tenantId: string }
  + Response: New User object
  + Used by: Add user modal
* **Update User**: PUT /users/{id}
  + Request: { name?: string, email?: string, role?: string, isActive?: boolean }
  + Response: Updated User object
  + Used by: Edit user functionality
* **Delete User**: DELETE /users/{id}
  + Response: Success message
  + Used by: User deletion functionality

**Tenant Management API**

**Tenant Operations**

* **List Tenants**: GET /tenants
  + Query params: Optional filtering and pagination
  + Response: { tenants: Tenant[], total: number, page: number, limit: number }
  + Used by: Tenant management dashboard
* **Get Tenant**: GET /tenants/{id}
  + Response: Tenant object with all details
  + Used by: Tenant detail view
* **Create Tenant**: POST /tenants
  + Request: { name: string, domain?: string, theme?: ThemeSettings, maxUsers?: number, maxFiles?: number, maxQueries?: number }
  + Response: New Tenant object
  + Used by: Create tenant form
* **Update Tenant**: PUT /tenants/{id}
  + Request: Tenant properties to update
  + Response: Updated Tenant object
  + Used by: Edit tenant functionality
* **Delete Tenant**: DELETE /tenants/{id}
  + Response: Success message
  + Used by: Tenant deletion functionality
* **Update Theme**: PUT /tenants/{id}/theme
  + Request: { theme: Partial<ThemeSettings> }
  + Response: Updated Tenant object
  + Used by: Theme customization interface

**Data Models**

**User Object**

json

Copy

{

"id": "string",

"email": "string",

"name": "string",

"role": "admin | contributor | reviewer | viewer",

"tenantId": "string"

}

**File Object**

json

Copy

{

"id": "string",

"name": "string",

"size": "number",

"type": "string",

"uploadDate": "string",

"lastModified": "string",

"source": "upload | google-drive | onedrive | dropbox | sharepoint",

"isExternal": "boolean",

"externalId": "string?",

"externalUrl": "string?",

"thumbnailUrl": "string?",

"tags": [

{

"id": "string",

"name": "string",

"color": "string?",

"confidence": "number?",

"approved": "boolean?",

"createdBy": "string?",

"approvedBy": "string?"

}

],

"categories": [

{

"id": "string",

"name": "string",

"color": "string?"

}

],

"createdBy": "string",

"status": "processing | indexed | error",

"errorMessage": "string?"

}

**Message Object**

json

Copy

{

"id": "string",

"conversationId": "string",

"content": "string",

"role": "user | assistant",

"timestamp": "string",

"sources": [

{

"fileId": "string",

"fileName": "string",

"snippet": "string",

"page": "number?"

}

]

}

**Conversation Object**

json

Copy

{

"id": "string",

"title": "string",

"createdAt": "string",

"updatedAt": "string",

"messages": "Message[]",

"fileContext": "string[]?"

}

**Tenant Object**

json

Copy

{

"id": "string",

"name": "string",

"domain": "string?",

"isActive": "boolean",

"createdAt": "string",

"updatedAt": "string",

"theme": {

"primaryColor": "string",

"logoUrl": "string?",

"favicon": "string?",

"darkMode": "boolean"

},

"maxUsers": "number?",

"maxFiles": "number?",

"maxQueries": "number?"

}

**API Error Handling**

**Error Response Format**

json

Copy

{

"error": {

"code": "string",

"message": "string",

"details": "any?"

}

}

**Common Error Codes**

* **Authentication Errors**:
  + 401: Unauthorized - Invalid or missing authentication
  + 403: Forbidden - Not permitted to access resource
* **Resource Errors**:
  + 404: Not Found - Resource doesn't exist
  + 409: Conflict - Resource already exists or state conflict
* **Validation Errors**:
  + 400: Bad Request - Invalid input
  + 422: Unprocessable Entity - Validation failed
* **Server Errors**:
  + 500: Internal Server Error - Unexpected server error
  + 503: Service Unavailable - Service temporarily unavailable

**Security Requirements**

**API Security Measures**

* **HTTPS**: All endpoints must be served over HTTPS
* **CORS**: Proper Cross-Origin Resource Sharing configuration
* **Rate Limiting**: Prevent abuse through appropriate rate limits
* **Input Validation**: Strict validation of all input data
* **Content Security Policy**: Appropriate headers for browser security
* **Authentication Timeout**: Token expiration and refresh mechanisms

**Sensitive Data Handling**

* **PII Protection**: Proper handling of Personally Identifiable Information
* **Data Encryption**: Encryption of sensitive data at rest and in transit
* **Logging Guidelines**: Avoid logging sensitive information
* **Token Security**: Secure token storage and transmission

**Implementation Considerations**

**Performance Recommendations**

* **Pagination**: All list endpoints should support pagination
* **Filtering**: Support for filtering large data sets
* **Response Time**: Target <500ms response time for typical requests
* **Caching**: Implement appropriate caching headers and mechanisms

**Versioning Strategy**

* **API Versioning**: Include version in URL path (e.g., /v1/files)
* **Backwards Compatibility**: Maintain compatibility within major versions
* **Deprecation Process**: Clear process for deprecating old endpoints

**Documentation**

* **OpenAPI/Swagger**: Maintain OpenAPI documentation
* **Example Requests/Responses**: Include examples for all endpoints
* **Error Documentation**: Document all possible error codes and messages