# x

## Overview

PingNotes is a comprehensive note-taking and file management system with role-based access control. The API supports three user roles: Admin, Teacher, and User, each with different permissions and capabilities.

Base URL: `http://localhost:5000` (or your configured server URL)

## Authentication

The API uses JWT (JSON Web Tokens) for authentication. Most endpoints require authentication via Bearer token in the Authorization header.

### Authentication Header Format

Authorization: Bearer <jwt\_token>

## User Roles & Permissions

|  |  |  |
| --- | --- | --- |
| Role | Description | Permissions |
| ------ | ------------- | ------------- |
| Admin | System administrator | Full access to all endpoints |
| Teacher | Educational content creator | Can manage subjects, topics, files, and notes |
| User | Student/learner | Can view and interact with content, upload files |

## API Endpoints

### 🔐 Authentication Routes

#### Google OAuth Authentication

GET `/api/auth/google/user`

• Description: Initiates Google OAuth flow for users

• Authentication: Not required

• Response: Redirects to Google OAuth consent screen

GET `/api/auth/google/user/callback`

• Description: Google OAuth callback for users

• Authentication: Not required

• Response: Redirects to frontend with JWT token

GET `/api/auth/google/teacher`

• Description: Initiates Google OAuth flow for teachers

• Authentication: Not required

• Response: Redirects to Google OAuth consent screen

GET `/api/auth/google/teacher/callback`

• Description: Google OAuth callback for teachers

• Authentication: Not required

• Response: Redirects to frontend with JWT token

POST `/api/auth/logout`

• Description: Logout endpoint

• Authentication: Not required

• Response:

{ "message": "Logged out successfully" }

### 👥 User Management

#### User Authentication

POST `/api/users/login`

• Description: Login for users with email/password

• Authentication: Not required

• Request Body:

{ "email": "user@example.com", "password": "password123" }

• Response:

{ "token": "jwt\_token\_here", "user": { "id": "user\_id", "name": "John Doe", "email": "user@example.com" }, "message": "Login successful" }

#### User CRUD Operations

GET `/api/users`

• Description: Get all users

• Authentication: Required (User role)

• Response: Array of user objects

GET `/api/users/:id`

• Description: Get user by ID

• Authentication: Required (User role)

• Parameters: `id` - User ID

• Response: User object

POST `/api/users`

• Description: Create new user

• Authentication: Required (User role)

• Request Body:

{ "name": "John Doe", "email": "john@example.com", "password": "password123" }

• Response: Created user object

PUT `/api/users/:id`

• Description: Update user

• Authentication: Required (User role)

• Parameters: `id` - User ID

• Request Body:

{ "name": "Updated Name", "email": "updated@example.com", "password": "newpassword" }

• Response:

{ "message": "User updated successfully", "user": { /\* updated user object \*/ } }

DELETE `/api/users/:id`

• Description: Delete user (soft delete)

• Authentication: Required (User role)

• Parameters: `id` - User ID

• Response:

{ "message": "User deleted successfully" }

### 👨‍🏫 Teacher Management

#### Teacher Authentication

POST `/api/teachers/login`

• Description: Login for teachers with email/password

• Authentication: Not required

• Request Body:

{ "email": "teacher@example.com", "password": "password123" }

• Response: Similar to user login with teacher role

#### Teacher CRUD Operations

GET `/api/teachers`

• Description: Get all teachers

• Authentication: Required (Teacher role)

• Response: Array of teacher objects

GET `/api/teachers/:id`

• Description: Get teacher by ID

• Authentication: Required (Teacher role)

• Parameters: `id` - Teacher ID

• Response: Teacher object

POST `/api/teachers`

• Description: Create new teacher

• Authentication: Required (Teacher role)

• Request Body:

{ "name": "Dr. Smith", "email": "smith@university.edu", "password": "password123" }

• Response: Created teacher object

PUT `/api/teachers/:id`

• Description: Update teacher

• Authentication: Required (Teacher role)

• Parameters: `id` - Teacher ID

• Request Body: Teacher fields to update

• Response: Updated teacher object

DELETE `/api/teachers/:id`

• Description: Delete teacher

• Authentication: Required (Teacher role)

• Parameters: `id` - Teacher ID

• Response: Success message

### 📚 Subject Management

GET `/api/subjects`

• Description: Get all subjects

• Authentication: Required (Admin, Teacher, User)

• Response: Array of subject objects with topics

GET `/api/subjects/:id`

• Description: Get subject by ID

• Authentication: Required (Admin, Teacher, User)

• Parameters: `id` - Subject ID

• Response: Subject object with populated topics

POST `/api/subjects`

• Description: Create new subject

• Authentication: Required (Admin, Teacher, User)

• Request Body:

{ "name": "Mathematics", "subject\_code": "MATH101" }

• Response: Created subject object

PUT `/api/subjects/:id`

• Description: Update subject

• Authentication: Required (Admin, Teacher, User)

• Parameters: `id` - Subject ID

• Request Body: Subject fields to update

• Response: Updated subject object

DELETE `/api/subjects/:id`

• Description: Delete subject

• Authentication: Required (Admin, Teacher, User)

• Parameters: `id` - Subject ID

• Response: Success message

### 📖 Topic Management

POST `/api/subjects/:subjectId/topics`

• Description: Add topic to subject

• Authentication: Required (Admin, Teacher, User)

• Parameters: `subjectId` - Subject ID

• Request Body:

{ "name": "Calculus", "description": "Introduction to calculus concepts" }

• Response: Created topic object

GET `/api/subjects/:subjectId/topics/:topicId`

• Description: Get topic by ID

• Authentication: Required (Admin, Teacher, User)

• Parameters:

• `subjectId` - Subject ID

• `topicId` - Topic ID

• Response: Topic object

PUT `/api/subjects/:subjectId/topics/:topicId`

• Description: Update topic

• Authentication: Required (Admin, Teacher, User)

• Parameters:

• `subjectId` - Subject ID

• `topicId` - Topic ID

• Request Body: Topic fields to update

• Response: Updated topic object

DELETE `/api/subjects/:subjectId/topics/:topicId`

• Description: Delete topic

• Authentication: Required (Admin, Teacher, User)

• Parameters:

• `subjectId` - Subject ID

• `topicId` - Topic ID

• Response: Success message

### 📁 File Management

POST `/api/files/upload`

• Description: Upload files (single or multiple)

• Authentication: Required (Admin, Teacher, User)

• Content-Type: `multipart/form-data`

• Request Body: Form data with files and optional fields:

• `files`: File(s) to upload

• `subject\_id`: (Optional) Subject ID to link file

• `topic\_id`: (Optional) Topic ID to link file

• Response:

{ "message": "2 file(s) uploaded successfully", "files": [ { "\_id": "file\_id", "name": "document.pdf", "mimetype": "application/pdf", "size": 1024000, "drive\_file\_id": "google\_drive\_file\_id", "drive\_file\_url": "https://drive.google.com/...", "uploaded\_by": "user\_id", "uploaded\_by\_role": "User", "linked\_subject": "subject\_id", "linked\_topic": "topic\_id", "created\_at": "2024-01-01T00:00:00.000Z" } ] }

GET `/api/files`

• Description: Get all files

• Authentication: Required (Admin, Teacher, User)

• Response: Array of file objects with populated references

GET `/api/files/:id`

• Description: Get file by ID

• Authentication: Required (Admin, Teacher, User)

• Parameters: `id` - File ID

• Response: File object with populated references

DELETE `/api/files/:id`

• Description: Delete file (removes from Google Drive and database)

• Authentication: Required (Admin, Teacher, User)

• Parameters: `id` - File ID

• Response: Success message

### 📝 Note Management

GET `/api/notes`

• Description: Get all notes

• Authentication: Required (Admin, Teacher, User)

• Response: Array of note objects with populated group and teacher references

GET `/api/notes/:id`

• Description: Get note by ID

• Authentication: Required (Admin, Teacher, User)

• Parameters: `id` - Note ID

• Response: Note object with populated references

POST `/api/notes`

• Description: Create new note

• Authentication: Required (Admin, Teacher, User)

• Request Body:

{ "title": "Lecture Notes", "description": "Important points from today's lecture", "group\_id": "group\_id", "teacher\_id": "teacher\_id" }

• Response: Created note object

PUT `/api/notes/:id`

• Description: Update note

• Authentication: Required (Admin, Teacher, User)

• Parameters: `id` - Note ID

• Request Body: Note fields to update

• Response: Updated note object

DELETE `/api/notes/:id`

• Description: Delete note

• Authentication: Required (Admin, Teacher, User)

• Parameters: `id` - Note ID

• Response: Success message

### 👥 Group Management

GET `/api/groups`

• Description: Get all groups

• Authentication: Not required

• Response: Array of group objects

POST `/api/groups`

• Description: Create new group

• Authentication: Not required

• Request Body: Group data

• Response: Created group object

### 🔧 Admin Management

POST `/api/admin/login`

• Description: Admin login

• Authentication: Not required

• Request Body:

{ "email": "admin@example.com", "password": "admin\_password" }

• Response: JWT token and admin data

GET `/api/admin`

• Description: Get all admins

• Authentication: Required (Admin role)

• Response: Array of admin objects

POST `/api/admin`

• Description: Create new admin

• Authentication: Required (Admin role)

• Request Body: Admin data

• Response: Created admin object

GET `/api/admin/:id`

• Description: Get admin by ID

• Authentication: Required (Admin role)

• Parameters: `id` - Admin ID

• Response: Admin object

PUT `/api/admin/:id`

• Description: Update admin

• Authentication: Required (Admin role)

• Parameters: `id` - Admin ID

• Request Body: Admin fields to update

• Response: Updated admin object

DELETE `/api/admin/:id`

• Description: Delete admin

• Authentication: Required (Admin role)

• Parameters: `id` - Admin ID

• Response: Success message

## Data Models

### User Model

{ "\_id": "ObjectId", "name": "String (required)", "email": "String (unique, required)", "password": "String (optional for Google OAuth)", "googleId": "String (unique, sparse)", "googleAccessToken": "String", "googleRefreshToken": "String", "profilePicture": "String", "created\_at": "Date", "updated\_at": "Date", "is\_active": "Boolean (default: true)", "is\_deleted": "Boolean (default: false)" }

### Subject Model

{ "\_id": "ObjectId", "name": "String (required)", "subject\_code": "String (unique)", "created\_by": "ObjectId (ref to User/Teacher)", "created\_by\_role": "String (enum: User, Teacher)", "topics": [ { "\_id": "ObjectId", "name": "String (required)", "description": "String", "files": ["ObjectId (ref to File)"], "created\_at": "Date" } ], "created\_at": "Date" }

### File Model

{ "\_id": "ObjectId", "name": "String (required)", "path": "String (local path, optional)", "mimetype": "String", "size": "Number", "drive\_file\_id": "String (Google Drive file ID)", "drive\_file\_url": "String (Google Drive public link)", "uploaded\_by": "ObjectId (ref to User/Teacher)", "uploaded\_by\_role": "String (enum: User, Teacher)", "linked\_subject": "ObjectId (ref to Subject, optional)", "linked\_topic": "ObjectId (ref to Subject.topics, optional)", "created\_at": "Date" }

### Note Model

{ "\_id": "ObjectId", "title": "String (required)", "description": "String", "group\_id": "ObjectId (ref to Group, required)", "teacher\_id": "ObjectId (ref to Teacher, required)", "created\_at": "Date" }

## Error Responses

### Standard Error Format

{ "message": "Error description", "error": "Error details (in development mode)" }

### Common HTTP Status Codes

|  |  |
| --- | --- |
| Status Code | Description |
| ------------- | ------------- |
| 200 | Success |
| 201 | Created |
| 400 | Bad Request - Invalid input |
| 401 | Unauthorized - Missing or invalid token |
| 403 | Forbidden - Insufficient permissions |
| 404 | Not Found - Resource not found |
| 409 | Conflict - Resource already exists |
| 500 | Internal Server Error |

## Google Drive Integration

The API integrates with Google Drive for file storage:

• Files are uploaded to Google Drive using the user's access token

• If user doesn't have Google access, files are uploaded to service account

• File URLs are stored for easy access

• Automatic token refresh handling for expired tokens

## Rate Limiting & Security

• JWT tokens expire after 1 day

• File uploads limited to 50MB per request

• CORS enabled for cross-origin requests

• Input validation on all endpoints

• Role-based access control on protected routes

## Development Notes

• Server runs on port 5000 by default

• MongoDB database required

• Google OAuth credentials needed for full functionality

• Environment variables required for JWT secret and Google credentials