Requirements Document for Knowledge Management System

1. Core Features (Basic Implementation)

1.1 User Authentication & Management

A basic user system for authentication and access control.
- Implement user registration, login, and logout (Django Allauth or JWT-based auth).
- Implement user roles (Student, Moderator, Admin).
- Profile management (name, avatar, bio, etc.).

1.2 Classroom System

A system for creating and managing public/private classrooms.
- Implement classroom creation with public/private visibility.
- Implement invitation system for private classrooms.
- Implement joining and leaving classrooms.
- Implement role-based moderation (ability to pin/delete messages, manage members).

1.3 Note-Taking System

A structured markdown-based note-taking system.
- Implement markdown editor for creating/editing notes.
- Implement the ability to attach images/audio to notes.
- Implement note sharing within classrooms.
- Implement version control for notes (history tracking).

2. Advanced Features (Enhancing Functionality)

2.1 AI-Powered Summarization

AI-based summarization for videos and notes.

- Implement YouTube transcript extraction using YouTube API.
- Process transcript with an LLM to generate structured summaries.
- Implement AI-generated summarization for user notes.

2.2 Collaborative Editing

Allow multiple users to edit notes in real-time.

- Implement real-time note updates using WebSockets/Django Channels.
- Implement user presence indicators (who is editing what).
- Implement conflict resolution for simultaneous edits.

2.3 Real-Time Communication

A chat system for student discussions.

- Implement Django Channels-based WebSocket chat.
- Implement message reactions (emoji support).
- Implement threaded discussions in classrooms.

2.4 Whiteboard Feature

An interactive whiteboard for real-time collaboration.

- Implement Canvas API for drawing support.
- Implement WebRTC for real-time P2P whiteboard updates.
- Implement WebSockets fallback when WebRTC is unavailable.

3. Scalability & Search Optimization

3.1 Full-Text Search

Implement a fast, scalable search system.

- Use PostgreSQL full-text search or Elasticsearch.
- Implement search across notes, discussions, and classrooms.
- Implement tagging and categorization for better organization.

3.2 Media Storage & Optimization

Efficient storage and retrieval of images and audio.

- Implement Django Storages with cloud support (S3/GCP Storage).
- Optimize uploaded media for performance.

4. Deployment & Security Considerations

4.1 Deployment Setup

A robust production-ready deployment system.

- Containerize the application using Docker.
- Set up Nginx as a reverse proxy.
- Use Gunicorn for serving the Django application.
- Deploy on AWS/GCP/VPS for scalability.

4.2 Security Best Practices

Ensuring application security.

- Implement CSRF, XSS, and SQL injection protection.
- Secure WebSockets communication.
- Implement access control policies for classrooms and notes.

5. Future Enhancements (Optional, Post-MVP)

5.1 Mobile API Support

```
A REST API for mobile and third-party integrations.
    - Implement DRF-based API endpoints.
    - Implement API authentication using JWT.
```

5.2 Advanced AI Features

```
Further enhancements using AI/ML.
    - Implement AI-powered quiz generation from notes.
    - Implement speech-to-text transcription for audio notes.
    - Implement personalized learning recommendations based on notes.
```

This structured requirements document ensures a clear development roadmap, starting with basic functionality and scaling up to advanced AI-driven features and real-time collaboration.