Study in Woods

Project Summary & Analysis

Team: Sahil Chouksey, Ajay Kori, Rohit Kumar Namdeo, Vivek Raj Dhurwey
Program: Master of Computer Applications (MCA) - Semester III
Institution: Gyan Ganga College of Technology, Jabalpur

1. Aim of the Project

Understanding the core objectives and motivation behind Study in Woods.

The primary aim of Study in Woods is to revolutionize the way MCA and similar program students approach academic learning by leveraging artificial intelligence and modern web technologies. The project addresses a critical gap in educational technology—the lack of intelligent, context-aware study assistance that understands course-specific content. Traditional learning management systems merely store documents; Study in Woods goes further by processing, understanding, and enabling intelligent conversations about academic materials.

The platform aims to provide students with a unified ecosystem where they can organize their academic resources, automatically extract structured information from syllabus PDFs, and receive AI-powered study guidance that cites specific course materials. By combining document management with conversational AI, the project seeks to reduce the cognitive load on students, enabling them to focus on understanding concepts rather than organizing materials.

2. What We Did & How We Did It

A comprehensive overview of the development process, technologies used, and implementation approach.

We developed a full-stack AI-powered educational platform from the ground up, implementing a three-tier microservices architecture. On the frontend, we built a responsive single-page application using Next.js 15 with React 19, incorporating modern UI patterns like server components, streaming, and optimistic updates. The backend was developed in Go using the Fiber framework, chosen for its exceptional performance characteristics—handling 100,000+ requests per second with minimal memory footprint. We implemented over 100 RESTful API endpoints covering authentication, academic hierarchy management, document processing, AI chat, and administrative functions.

For the AI capabilities, we integrated DigitalOcean's GradientAI platform, utilizing the Llama 3.3 70B model for natural language understanding and generation. We implemented a Retrieval-Augmented Generation (RAG) pipeline where uploaded documents are processed, chunked, and indexed into knowledge bases. When students ask questions, the system retrieves relevant context from these knowledge bases and generates accurate, citation-backed responses streamed in real-time via Server-Sent Events (SSE).

Document processing involved building an intelligent pipeline that accepts PDF uploads, extracts text content, and uses AI to parse syllabus structures—automatically identifying units, topics, and subtopics. For scanned documents, we developed a Python FastAPI microservice with OCR capabilities. The entire application was containerized using Docker and deployed on DigitalOcean infrastructure with PostgreSQL for persistent storage, Redis for caching and session management, and DigitalOcean Spaces for file storage.

3. Database Design

> *The data architecture powering the platform's functionality.*

The database architecture follows a carefully normalized schema (Third Normal Form) implemented in PostgreSQL 15.x, comprising over 30 tables organized into logical domains. The core academic hierarchy follows a one-to-many cascade: Universities contain Courses, which contain Semesters, which contain Subjects. This structure mirrors real-world academic organization and enables efficient querying at any level.

User management implements JWT-based authentication with a token blacklist table for secure logout and session invalidation. The document management system tracks uploaded files through their lifecycle—from upload to processing to indexing—with separate tables for raw documents, extracted syllabi, syllabus units, and individual topics. This granular structure enables precise content retrieval during AI conversations.

The chat system employs a sophisticated memory management architecture. Chat sessions are linked to specific subjects, enabling context-aware responses. Chat messages store both user queries and AI responses along with citation metadata pointing to specific document chunks. To optimize token usage with the LLM, we implemented chat memory compaction—periodically summarizing older messages while preserving essential context in a separate compacted_contexts table. All tables include audit columns (created_at, updated_at, deleted_at) with soft delete support, and foreign key constraints with appropriate cascade rules ensure referential integrity across the entire schema.

4. Design Choices & Architecture

> *The technical decisions that shaped the platform's structure and performance.*

Our architectural decisions were driven by three priorities: performance, maintainability, and scalability. We chose Go for the backend due to its compiled nature, excellent concurrency model (goroutines), and minimal runtime overhead—critical for handling real-time AI streaming to potentially thousands of concurrent users. The Fiber framework was selected for its Express.js-like simplicity while delivering performance comparable to raw net/http.

The frontend stack of Next.js 15 with React 19 was chosen for its hybrid rendering capabilities—static generation for marketing pages, server-side rendering for SEO-critical content, and client-side rendering for interactive dashboards. TanStack Query manages server state with automatic caching, background refetching, and optimistic updates, providing a snappy user experience even on slower connections.

We implemented several design patterns to ensure clean code architecture: the Repository Pattern abstracts database operations behind interfaces, enabling easy testing and potential database migration; the Service Layer separates business logic from HTTP handlers; Middleware Chain pattern handles cross-cutting concerns like authentication, rate limiting, and logging; and the Factory Pattern with dependency injection enables flexible service initialization and testing.

For AI integration, we adopted a RAG (Retrieval-Augmented Generation) architecture rather than fine-tuning. This approach allows the system to provide accurate, up-to-date responses based on actual course materials without the cost and complexity of model training. Documents are chunked using semantic splitting, embedded into vector representations, and stored in DigitalOcean's Knowledge Bases for efficient similarity search during query processing.

5. What It Accomplishes

The tangible outcomes and capabilities delivered by the platform.

Study in Woods successfully delivers an end-to-end AI-powered learning platform that transforms how students interact with academic content. The platform accomplishes several key objectives that directly benefit students and administrators alike.

Intelligent Document Processing: Students can upload syllabus PDFs, and the system automatically extracts structured information—course objectives, units, topics, and subtopics—eliminating hours of manual organization. The extraction accuracy exceeds 90% for well-formatted documents.

Context-Aware AI Assistance: Unlike generic chatbots, Study in Woods provides subject-specific AI assistance that understands the exact curriculum. When a student asks about a topic, the AI retrieves relevant sections from uploaded materials and generates responses with citations pointing to specific documents and page numbers.

Unified Academic Organization: The hierarchical structure of Universities, Courses, Semesters, and Subjects provides intuitive organization that mirrors real academic programs. Students can quickly navigate to any subject and access all related materials, chat history, and extracted syllabus content.

Real-Time Streaming Responses: AI responses are streamed token-by-token using SSE, providing immediate feedback and natural conversational flow rather than waiting for complete response generation.

Administrative Control: Administrators have full control over academic content management, system configuration, and audit logging, ensuring the platform can be managed effectively at institutional scale.

6. Future Scope

Planned enhancements and the roadmap for platform evolution.

The current implementation, while functional and feature-complete, relies on DigitalOcean's managed Knowledge Bases for vector storage and retrieval. This dependency introduces rate limits and scalability constraints that could become limiting factors as the user base grows. **In future iterations, we plan to migrate to a self-hosted vector database solution such as Qdrant, Milvus, or Weaviate.** This transition will provide several advantages: unlimited storage capacity constrained only by infrastructure, custom embedding models optimized for academic content, finer control over indexing and retrieval parameters, and elimination of per-request API costs. The self-hosted approach will enable horizontal scaling across multiple nodes, supporting millions of document chunks and thousands of concurrent similarity searches.

A significant planned enhancement is the democratization of content creation. Currently, academic content is managed centrally, but we envision opening the platform for any user to create their own courses, semesters, and subjects. Users would be able to upload their own syllabi, previous year question papers (PYQs), notes, and reference books, creating personalized learning environments tailored to their specific curriculum—even for institutions not yet in the system.

To sustain this expanded functionality, we will implement a subscription-based financial model. Users will be charged a monthly fee for each subject they create, with tiered limits on storage (e.g., 500MB per subject), document count (e.g., 50 documents), and AI query volume (e.g., 1000 queries/month). Premium tiers will offer increased limits, priority AI processing, and advanced analytics. This model ensures sustainable infrastructure costs while keeping the platform accessible—students can start with one or two subjects at minimal cost and expand as needed.

Additional future enhancements include: mobile applications for iOS and Android, collaborative study groups with shared knowledge bases, integration with university LMS platforms via LTI, advanced analytics with learning pattern recognition, and support for multimedia content including video lectures with automatic transcription and indexing.

7. Conclusion

Reflecting on the project's achievements and its potential impact.

Study in Woods represents a successful implementation of AI-powered educational technology, demonstrating how modern web development practices, cloud infrastructure, and large language models can be combined to create meaningful learning tools. The project achieves its core objectives of simplifying academic organization, enabling intelligent content extraction, and providing context-aware AI assistance for students.

Through careful architectural decisions—choosing Go for performance, implementing RAG for accurate responses, and designing a normalized database schema—we built a platform capable of handling real-world academic workloads. The three-tier architecture ensures separation of concerns, making the codebase maintainable and extensible. The integration of streaming AI responses provides a modern, responsive user experience that feels natural and immediate.

The platform validates the hypothesis that AI can meaningfully enhance education when properly contextualized. Generic AI assistants often provide incorrect or irrelevant information for academic queries; by grounding responses in actual course materials and providing citations, Study in Woods ensures students receive accurate, verifiable assistance. This approach builds trust and encourages adoption.

Looking forward, the transition to self-hosted vector databases and the introduction of user-generated content with a sustainable financial model will transform Study in Woods from a standalone application into a scalable educational ecosystem. The foundation built in this project—robust APIs, efficient data models, and proven AI integration patterns—positions the platform for this evolution. Study in Woods demonstrates that with thoughtful design and modern technology, we can build educational tools that genuinely improve how students learn, making quality AI-assisted education accessible to all.