Study in Woods

AI-Powered Study Companion Platform

MCA III Semester Minor Project | GGCT, Jabalpur | Session 2025-26

Team: Sahil Chouksey, Siddarth Verma, Anupama

What is Study in Woods?

Study in Woods is a web application that helps university students organize their study materials and get AI-powered assistance. The platform allows students to:

- Upload syllabus PDFs and automatically extract structured content (units, topics, books)
- Upload Previous Year Question papers and extract questions with metadata
- Chat with an AI assistant that understands their course materials
- Organize materials by University → Course → Semester → Subject

Technology Stack

Frontend (What Users See)

Technology	Version	Purpose
Next.js	15.5.6	React framework with App Router
React	19.1.0	UI library
TypeScript	5.x	Type-safe JavaScript
Tailwind CSS	4.x	Utility-first styling
shadcn/ui	new-york	Pre-built UI components (Radix UI)

TanStack Query	5.90.9	Server state management
Axios	1.13.2	HTTP client for API calls
Framer Motion	12.23.24	Animations

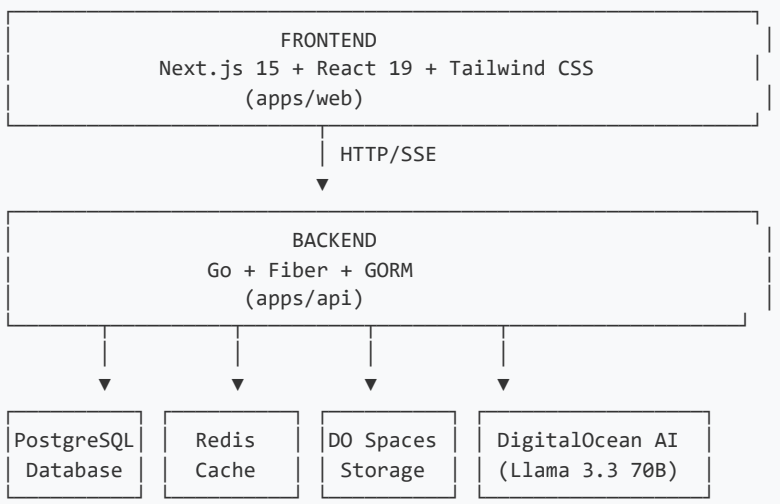
Backend (The Server)

Technology	Version	Purpose
Go (Golang)	1.24	Programming language
Fiber	v2	Web framework
GORM	latest	Database ORM
PostgreSQL	15	Primary database
Redis	7	Caching & session storage

Cloud Services (DigitalOcean)

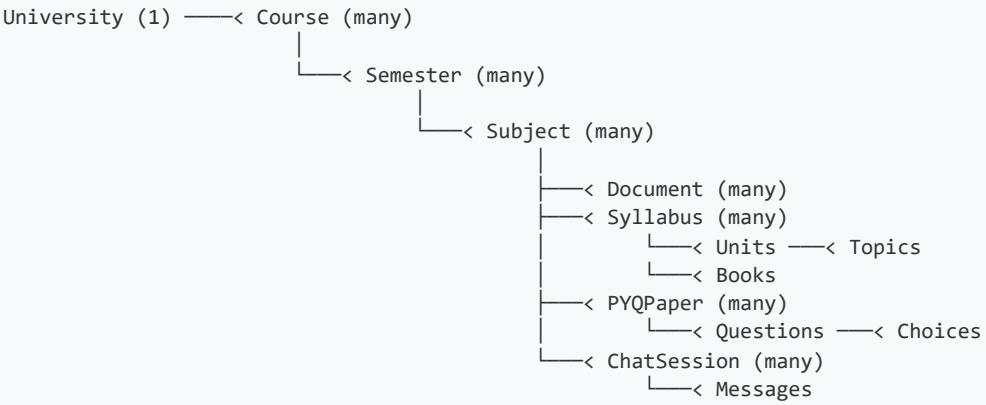
Service	Purpose
DigitalOcean Spaces	File storage (S3-compatible)
DigitalOcean Inference API	AI model access (Llama 3.3 70B)
DigitalOcean GenAI	Knowledge bases & AI agents

Architecture Overview



Database Schema

Entity Relationship Summary



Main Tables (21 total)

Table	Purpose
users	User accounts with email, password hash, role

universities	University list (RGPV, etc.)
courses	Courses (MCA, BCA, etc.)
semesters	Semester numbers per course
subjects	Subjects with AI agent/knowledge base IDs
documents	Uploaded files (PDFs, docs)
syllabuses	Extracted syllabus metadata
syllabus_units	Units within a syllabus
syllabus_topics	Topics within units
book_references	Textbook/reference recommendations
pyq_papers	Previous year question papers
pyq_questions	Individual questions
pyq_question_choices	Multiple choice options
chat_sessions	AI chat sessions per subject
chat_messages	Chat message history
jwt_token_blacklist	Revoked JWT tokens
admin_audit_logs	Admin action logging
app_settings	Application configuration
cron_job_logs	Scheduled job logs
api_key_usage_logs	External API key tracking
course_payments	Payment records (future)

API Endpoints Summary

Total Endpoints: ~100

Public Endpoints

Method	Endpoint	Purpose
GET	/ping	Health check
POST	/api/v1/auth/register	User registration
POST	/api/v1/auth/login	User login
GET	/api/v1/universities	List universities
GET	/api/v1/courses	List courses

Protected Endpoints (Require JWT)

Method	Endpoint	Purpose
GET	/api/v1/profile	Get user profile
POST	/api/v1/subjects/:id/documents	Upload document
POST	/api/v1/chat/sessions	Create chat session
POST	/api/v1/chat/sessions/:id/messages	Send message

SSE Streaming (Real-time)

Method	Endpoint	Purpose
GET	/api/v2/documents/:id/extract-syllabus?stream=true	Stream extraction progress

Key Features

1. AI-Powered Syllabus Extraction

> **How it works:**
>
> 1. User uploads PDF to DigitalOcean Spaces
> 2. Backend downloads PDF and extracts text
> 3. Sends text to Llama 3.3 70B model via DigitalOcean Inference API
> 4. AI returns structured JSON with units, topics, hours, books
> 5. Data saved to database
> 6. Real-time progress via Server-Sent Events (SSE)

AI Model Configuration:

- Model: `llama3.3-70b-instruct`
- API: `https://inference.do-ai.run/v1/chat/completions`
- Max Tokens: 16,384
- Temperature: 0.1 (for consistent output)

2. PYQ (Previous Year Questions) Extraction

- Extract questions from exam PDFs
- Detect sections, marks, unit numbers
- Handle "OR" questions (answer any one)
- Search external sources (RGPV Online)
- One-click ingestion of external papers

3. AI Chat Assistant

- Subject-specific chat sessions
- Uses DigitalOcean GenAI agents
- Context from uploaded documents
- Message history preserved

Authentication & Security

Feature	Implementation
Password Hashing	bcrypt (cost 12)
Authentication	JWT (access + refresh tokens)
Access Token Expiry	24 hours
Refresh Token Expiry	7 days
Brute Force Protection	Redis-backed, progressive lockout
Rate Limiting	IP-based (Fiber middleware)
Token Revocation	Database blacklist + version tracking

File Storage

Feature	Details
Provider	DigitalOcean Spaces (S3-compatible)
Max File Size	50 MB
Supported Types	PDF, DOCX, TXT, XLSX, PPTX, HTML, JSON
Download	Presigned URLs (60 min expiry)

Frontend Pages

Route	Purpose	Auth Required
/	Landing page with FAQ	No
/login	User login	No

/register	User registration	No
/dashboard	Main app interface	Yes
/chat	AI chat	No
/courses	Course selection	No
/history	Chat history	No
/settings	User settings	No

Dashboard Tabs

1. **Chat** - AI assistant interface
2. **Courses** - University/Course/Semester/Subject selection
3. **History** - Past chat sessions
4. **Settings** - Profile & preferences

Project Structure

```
study-in-woods/
├── apps/
│   ├── api/                    # Go Backend
│   │   ├── handlers/           # HTTP request handlers
│   │   ├── services/           # Business logic
│   │   ├── model/              # Database models
│   │   ├── database/           # DB connection & migrations
│   │   ├── utils/              # Utilities (auth, cache, validation)
│   │   ├── router/             # API routes
│   │   └── config/             # Configuration
│   │
│   ├── web/                    # Next.js Frontend
│   │   ├── src/
│   │   │   ├── app/            # Next.js pages (App Router)
│   │   │   ├── components/     # React components
│   │   │   ├── lib/api/        # API client & hooks
│   │   │   ├── providers/      # Context providers
│   │   │   └── types/          # TypeScript types
│   │   └── public/             # Static assets
│   │
│   └── ocr-service/            # Python OCR microservice
│
├── docker-compose.yml          # Local development containers
└── PROJECT_OVERVIEW.md         # Documentation
```

Workflows

Syllabus Upload Flow

1. User selects semester
2. Uploads PDF file
3. Frontend → POST /api/v2/semesters/:id/syllabus/upload
4. Backend uploads to Spaces
5. Returns document_id
6. Frontend connects to SSE stream
7. Backend downloads PDF, extracts text
8. Sends to Llama 3.3 70B
9. Parses JSON response
10. Creates subjects, units, topics in DB

11. Streams progress events to frontend
12. User sees extracted content

Chat Flow

1. User opens subject
2. Creates/selects chat session
3. Types message
4. Frontend → POST /api/v1/chat/sessions/:id/messages
5. Backend loads message history
6. Calls DigitalOcean Agent API
7. AI responds with context from documents
8. Message saved to database
9. Response displayed to user

Cron Jobs

Schedule	Job	Purpose
Every 15 min	Check Document Indexing	Update indexing status from DO
Every 30 min	Cleanup Pending Uploads	Remove stuck uploads
Every hour	Aggregate Statistics	Calculate usage stats
Daily 2 AM	Cleanup Old Data	Remove expired tokens, old logs

Security Features

1. **JWT with JTI tracking** - Individual token revocation
2. **Token versioning** - Mass session invalidation
3. **bcrypt password hashing** - Cost factor 12
4. **Progressive lockouts** - 2min → 1hr → 24hr
5. **Security headers** - XSS, HSTS, X-Frame-Options (Helmet)
6. **CORS configuration** - Configurable origins
7. **Input validation** - go-playground/validator
8. **Admin audit logging** - All admin actions tracked

What Makes This Project Special

1. **Real AI Integration** - Uses actual Llama 3.3 70B model, not mock data
2. **Streaming Progress** - Real-time extraction updates via SSE
3. **Structured Extraction** - AI outputs structured JSON, not just text
4. **Knowledge Base Integration** - Documents indexed for RAG queries
5. **Modern Stack** - Next.js 15, React 19, Go 1.24, Tailwind v4
6. **Production Ready** - Docker, health checks, cron jobs, error handling

Team Contributions

Team Member	Responsibilities
Sahil Chouksey	Full-stack development, AI integration, Architecture
Siddarth Verma	Backend development, Database design
Anupama	Frontend development, UI/UX

Quick Reference

Component	Location	Tech
Frontend	apps/web	Next.js 15, React 19, Tailwind
Backend	apps/api	Go, Fiber, GORM
Database	PostgreSQL	21 tables
Cache	Redis	Brute force, jobs
Storage	DO Spaces	PDF, documents
AI	DO Inference	Llama 3.3 70B
OCR	apps/ocr-service	Python, FastAPI

Built for: University students who want to study smarter
Built with: Modern web technologies and AI
Repository: study-in-woods