A

*MINOR PROJECT REPORT ON*

# Study in Woods

Submitted to the
**RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL**

In partial fulfillment of the requirement for the award of the Degree of
**MASTER OF COMPUTER APPLICATIONS**

Submitted to
**DEPARTMENT OF COMPUTER APPLICATIONS**
**Gyan Ganga College of Technology**
**Jabalpur (M.P.)**

## Under the guidance of


**Dr. Meghna Utmal**
**Prof. & Head MCA**

**Mr. Varun Garg**
**Prof. MCA**


## Submitted by


**SAHIL CHOUKSEY (0208CA241050)**
**ROHIT KUMAR NAMDEO (0208CA241048)**
**VIVEK RAJ DHURWEY (0208CA241071)**
**AJAY KORI (0208CA241009)**


**MCA III SEM | SESSION 2025-26**

# ACKNOWLEDGEMENT

Behind every successful effort there lies contribution from numerous sources irrespective of their magnitude, my project has no exception and I take this opportunity to thank those helping hand whole heartedly.

First and foremost, we have great pleasure in expressing my deep sense of gratitude to all the **Management Members,** Gyan Ganga Group of Institutions, for providing me the needed facilities throughout the duration of the project.

We are especially thankful to **Dr AjayLala,** Director Principal, Gyan Ganga College of Technology, for his continuous guidance and support throughout the duration of the project.

We would like to owe my gratitude to **Dr. Meghna Utmal** HOD, Department of Master of Computer Applications (MCA), for providing us a platform to initiate out towards the degree of computer application.

At last but not the least my heartily recognition to all our academic teachers, my family members, friends and those who directly or indirectly helped me in this endeavor.

**Thanks & Regards**

***Name of Students***

1. Sahil Chouksey
2. Rohit Kumar Namdeo
3. Vivek Raj Dhurwey
4. Ajay Kori


This is to certify that the project report entitled "**Study in Woods**" which has been completed and submitted by ***Sahil Chouksey, Rohit Kumar Namdeo, Vivek Raj Dhurwey, Ajay Kori*** **III SEM MCA** the Student of Master of Computer Applications (MCA) is a bonafide work by his/her. This Project report has been approved by us. The project report is satisfactory both in respect of its contents and literally representation.

This project report is in accordance with the requirement of degree of Master of Computer Applications (MCA) awarded by RAJIV GANDHI PROUDYOGIKI VISHWAVIDALAYA, Bhopal (M.P.).

**Dr. Meghna Utmal**
**HOD (MCA)**
**GGCT, Jabalpur**


This is to certify that the project report entitled "**Study in Woods**" which has been completed and submitted by ***Sahil Chouksey, Rohit Kumar Namdeo, Vivek Raj Dhurwey, Ajay Kori*** **III SEM MCA** the Student of Master of Computer Applications (MCA) is a bonafide work by his/her. This Project report has been approved by us. The project report is satisfactory both in respect of its contents and literally representation.

This project report is in accordance with the requirement of degree of Master of Computer Applications (MCA) awarded by RAJIV GANDHI PROUDYOGIKI VISHWAVIDALAYA, Bhopal (M.P.).

**Internal Examiner:**
**Date:**

**External Examiner:**
**Date:**


# CANDIDATE DECLARATION

I hereby declare that this project report titled **"Study in Woods"** submitted by me in fulfillment for the award of Master of Computer Applications (M.C.A.) by Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal is a result of authentic work under taken by me.

The same has not been submitted by me to this or any other university for any other graduate/post graduate course whatsoever.

**Project Submitted By:**

***Name of Students***

1. Sahil Chouksey
2. Rohit Kumar Namdeo
3. Vivek Raj Dhurwey
4. Ajay Kori

# TABLE OF CONTENTS

# 1. INTRODUCTION

## 1.1 About the Project

**Study in Woods** is an AI-powered educational platform that helps students organize academic materials and interact with course content intelligently. The system processes syllabus PDFs using NLP to extract structured information and provides an AI chat interface for study guidance.

## 1.2 Purpose of the Project

The primary objectives are:

- **Academic Organization:** Centralized platform for managing universities, courses, semesters, and subjects
- **Intelligent Content Extraction:** AI-powered processing of syllabus PDFs with automatic data extraction
- **AI-Assisted Learning:** Conversational AI for answering questions and providing study guidance
- **Exam Preparation:** Past year question paper extraction and categorization linked to syllabus topics
- **Progress Tracking:** Study progress monitoring and personalized study plan management

## 1.3 System Architecture

The application follows a three-tier microservices architecture:

- **Frontend:** Next.js 15 with React 19 providing responsive UI
- **Backend:** Go (Golang) with Fiber framework handling API and business logic
- **Data Layer:** PostgreSQL for primary storage, Redis for caching and sessions

- **OCR Service:** Python FastAPI microservice for text extraction from scanned documents

## 1.4 Core Functionalities

1. **User Management:** Secure authentication with role-based access (Admin/Student)
2. **Document Processing:** Syllabus upload with AI-powered content extraction
3. **AI Chat Interface:** Subject-specific conversational assistance
4. **PYQ Management:** Question paper extraction and categorization
5. **Analytics Dashboard:** Usage statistics and performance tracking
6. **RESTful API:** 100+ endpoints for programmatic access

## 1.5 User Characteristics

**Students:** MCA or similar program students seeking AI-assisted learning and exam preparation tools.

**Administrators:** Academic staff managing universities, courses, and system settings with audit capabilities.

## 1.6 Operating Environment

- **Client:** Modern web browsers (Chrome, Firefox, Safari, Edge) on all devices
- **Server:** Docker containerized deployment on Linux with HTTPS
- **Cloud:** DigitalOcean Spaces for storage, GradientAI (Llama 3.3 70B) for AI features

## 1.7 Constraints and Dependencies

**Constraints:** Go/Next.js stack required; PDF-only uploads; must support 1000+ concurrent users.

**Dependencies:** DigitalOcean services (AI, storage), PostgreSQL, Redis, third-party libraries.

# 2. PROJECT UNDERSTANDING DOCUMENT

## 2.1 Problem Statement

In the contemporary educational landscape, students pursuing higher education, particularly in technical fields like Master of Computer Applications (MCA), encounter several critical challenges in managing their academic materials and optimizing their learning processes:

- **Information Overload:** Students receive voluminous course syllabi in PDF format containing complex hierarchical information about course units, topics, and learning outcomes, making manual extraction and organization time-consuming and error-prone.
- **Fragmented Resources:** Academic materials are scattered across multiple platforms - university portals, email attachments, physical documents, and personal storage, leading to inefficient access and retrieval.
- **Limited Study Assistance:** Traditional learning methods lack personalized, on-demand guidance for clarifying doubts and understanding complex topics outside of classroom hours.
- **Exam Preparation Challenges:** Students struggle to systematically organize and categorize previous year questions, making targeted exam preparation difficult.
- **Lack of Progress Tracking:** Absence of integrated tools to monitor study progress, set goals, and manage academic todos results in suboptimal time management.

## 2.2 Proposed Solution

**Study in Woods** addresses these challenges through an integrated, AI-powered platform that combines intelligent document processing, conversational AI, and comprehensive academic management capabilities.

### 2.2.1 Solution Components

#### Component 1: Intelligent Syllabus Processing

- Automated PDF parsing using advanced NLP techniques
- Hierarchical extraction of units, topics, and subtopics
- Structured storage for easy navigation and retrieval
- Support for multiple syllabus formats across universities

#### Component 2: AI-Powered Study Assistant

- Context-aware chatbot powered by RAG (Retrieval Augmented Generation)
- Subject-specific knowledge bases for accurate responses
- Natural language query processing for intuitive interaction
- Citation of source materials for verifiable answers

#### Component 3: Academic Organization System

- University → Course → Semester → Subject hierarchy management
- Document upload and categorization capabilities
- Previous year question paper organization
- Personal notes and resource management

> Component 4: Progress Tracking Dashboard
>
> - Visual analytics for study patterns and progress
> - Goal setting and achievement tracking
> - Todo management for academic tasks
> - Performance insights and recommendations

## 2.3 Project Scope

### 2.3.1 In Scope

| Category | Features Included |
|---|---|
| **User Management** | Registration, authentication, profile management, role-based access (Student/Admin) |
| **Academic Structure** | University, course, semester, and subject management with full CRUD operations |
| **Document Processing** | PDF upload, syllabus extraction, document storage, and retrieval |
| **AI Chat System** | Subject-specific chatbot, conversation history, RAG-based responses |
| **Analytics** | Usage statistics, study patterns, progress visualization |
| **Administration** | User management, system configuration, audit logging |

### 2.3.2 Out of Scope

- Mobile native applications (iOS/Android) - Web responsive design provided instead
- Offline functionality - Requires internet connectivity
- Video content processing - Focus on PDF and text documents
- Real-time collaboration features - Individual study focus
- Payment/subscription management - Free platform for students

## 2.4 Target Users

### 2.4.1 Primary Users

**Students:** MCA and similar postgraduate program students who need:

- Organized access to course syllabi and materials
- AI-assisted study guidance and doubt resolution
- Efficient exam preparation tools
- Progress tracking and goal management

### 2.4.2 Secondary Users

**Administrators:** College/university staff responsible for:

- Managing academic structure (universities, courses, subjects)
- Uploading and maintaining syllabus documents
- Monitoring platform usage and user activity
- System configuration and maintenance

## 2.5 Expected Outcomes

| Outcome | Measurement Criteria |
| --- | --- |
| **Improved Study Efficiency** | Reduced time spent searching for academic materials by 60% |
| **Enhanced Understanding** | AI chat resolves 80% of student queries without external help |
| **Better Organization** | All academic materials accessible from single platform |
| **Exam Readiness** | Systematic coverage of syllabus topics with progress tracking |
| **Time Savings** | Automated syllabus extraction saves 2+ hours per subject |

## 2.6 Constraints and Assumptions

### 2.6.1 Constraints

- **Technical:** Dependent on DigitalOcean AI platform availability and API limits
- **Data:** Syllabus extraction accuracy depends on PDF formatting consistency
- **Performance:** AI response generation limited by LLM processing time
- **Storage:** Cloud storage costs scale with document uploads

### 2.6.2 Assumptions

- Users have reliable internet connectivity
- Syllabus PDFs follow standard academic formatting conventions
- Users possess basic computer literacy skills

- University academic structures follow hierarchical organization
- English is the primary language for academic content

# 3. REQUIREMENTS

## 3.1 Functional Requirements

| ID | Requirement | Description |
|---|---|---|
| FR-1 | User Registration | Email/password signup with validation and duplicate prevention |
| FR-2 | JWT Authentication | Secure token-based login with Redis session management |
| FR-3 | Role-Based Access | Student and Admin roles with differentiated permissions |
| FR-4 | Academic Hierarchy | University > Course > Semester > Subject management |
| FR-5 | Document Upload | PDF upload (10MB max) to DigitalOcean Spaces storage |
| FR-6 | AI Syllabus Extraction | Auto-extract units, topics from syllabus PDFs (85% accuracy) |
| FR-7 | Document Indexing | Auto-index documents into subject-specific AI knowledge bases |
| FR-8 | AI Chat Interface | Llama 3.3 70B powered conversational assistant with context |
| FR-9 | PYQ Management | Extract and categorize past year questions by topic/difficulty |
| FR-10 | API Key Management | Encrypted API keys with scopes, rate limiting, and expiration |

## 3.2 Non-Functional Requirements

| ID | Category | Requirement |
|---|---|---|
| NFR-1 | Performance | 95% requests under 2s; AI responses stream within 5s |
| NFR-2 | Scalability | 1000+ concurrent users; 10,000 requests/minute capacity |
| NFR-3 | Security | JWT RS256, bcrypt hashing, AES-256 encryption, HTTPS enforced |
| NFR-4 | Availability | 99.5% uptime with graceful failure handling |
| NFR-5 | Usability | Responsive design (desktop/tablet/mobile), WCAG 2.1 AA compliant |
| NFR-6 | Reliability | ACID compliance, daily backups, 30-day point-in-time recovery |
| NFR-7 | Maintainability | Clean architecture, 70% test coverage, comprehensive logging |
| NFR-8 | Protection | Rate limiting, CORS, input validation against SQL injection/XSS |

## 3.3 Hardware Requirements

### *3.3.1 Client-Side*

| Component | Minimum | Recommended |
|---|---|---|
| Processor | Dual-core 1.6 GHz | Quad-core 2.4 GHz |
| RAM | 2 GB | 4 GB |
| Network | 1 Mbps | 5 Mbps broadband |
| Browser | Chrome 90+, Firefox 88+, Safari 14+, Edge 90+ | |

### 3.3.2 Server-Side

| Component | App Server | Database | Cache |
|---|---|---|---|
| CPU | 4 vCPU | 4 vCPU | 2 vCPU |
| RAM | 8 GB | 8 GB | 4 GB |
| Storage | 80 GB SSD | 200 GB SSD | 40 GB SSD |
| OS | Ubuntu 22.04 LTS | | |

## 3.4 Software Requirements

### 3.4.1 Frontend Stack

| Technology | Version |
|---|---|
| Next.js | 15.5.6 |
| React | 19.1.0 |
| TypeScript | 5.x |
| Tailwind CSS | 4.0 |
| TanStack Query | 5.90.9 |

### 3.4.2 Backend Stack

| Technology | Version |
|---|---|
| Go (Golang) | 1.24.1 |
| Fiber | 2.52.5 |
| GORM | 1.31.0 |
| PostgreSQL | 15.x |
| Redis | 7.x |

### 3.4.3 Cloud Services

| Service | Provider |
|---|---|
| Compute (Droplets) | DigitalOcean |
| Object Storage (Spaces) | DigitalOcean |
| AI Platform (Llama 3.3 70B) | DigitalOcean GradientAI |
| Knowledge Bases | DigitalOcean AI |

## 3.5 External Interface Requirements

- **User Interface:** Responsive web UI with dashboard, document upload, AI chat, and admin panels
- **API Interface:** RESTful JSON API with JWT authentication, rate limiting, and OpenAPI documentation
- **AI Platform:** DigitalOcean GradientAI API for chat completions and knowledge base management

# 4. TECHNOLOGY USED

## 4.1 Technology Stack Overview

The Study in Woods platform uses a modern, scalable technology stack with clear separation between frontend, backend, database, and cloud services layers.

## 4.2 Frontend Technologies

| Technology | Version | Purpose | Key Features |
|---|---|---|---|
| Next.js | 15.5.6 | React Framework | SSR, SSG, API Routes, Turbopack, Image Optimization |
| React | 19.1.0 | UI Library | Server Components, Suspense, Virtual DOM, Hooks |
| TypeScript | 5.x | Type Safety | Static Typing, IntelliSense, Compile-time Errors |
| Tailwind CSS | 4.0 | Styling | Utility Classes, JIT Compiler, Dark Mode, Responsive |
| shadcn/ui | Latest | UI Components | Accessible, Customizable, Radix UI Primitives |
| TanStack Query | 5.90.9 | State Management | Caching, Auto-refetch, Optimistic Updates |
| Framer Motion | 12.23.24 | Animations | Declarative Animations, Gestures, Layout Transitions |
| React Hook Form | 7.66.0 | Form Management | Validation, Performance, Error Handling |
| Zod | 4.1.12 | Schema Validation | Type-safe Schemas, Runtime Validation |
| Axios | 1.13.2 | HTTP Client | Interceptors, Request Cancellation, Auto JSON |

## 4.3 Backend Technologies

| Library | Version | Purpose | Key Capabilities |
|---|---|---|---|
| Go | 1.24.1 | Programming Language | Goroutines, Channels, Fast Compilation, GC |
| Fiber | 2.52.5 | Web Framework | Fasthttp, Middleware, Routing, Context |
| GORM | 1.31.0 | ORM | Migrations, Associations, Hooks, Preloading |
| JWT | 5.3.0 | Authentication | Token Generation, RS256, Claims Validation |
| bcrypt | 0.43.0 | Password Hashing | Adaptive Cost, Automatic Salting |
| go-redis | 9.16.0 | Redis Client | Connection Pooling, Pipelining, Pub/Sub |
| AWS SDK | 1.55.8 | S3 Client | Multipart Upload, Pre-signed URLs, Retries |
| Validator | 10.28.0 | Input Validation | Struct Tags, Custom Validators, Error Messages |
| Cron | 3.0.1 | Job Scheduling | Cron Expressions, Job Chains, Error Handling |

## 4.4 Database Technologies

| Technology | Version | Type | Primary Use Cases |
|---|---|---|---|
| PostgreSQL | 15.x | Relational Database | Permanent data storage, Complex queries, ACID transactions, JSONB support |
| Redis | 7.x | In-Memory Cache | Session storage, Rate limiting, Temporary data, Pub/Sub |

## 4.5 Cloud Services & Infrastructure

DigitalOcean provides the complete cloud infrastructure, selected for its simplicity, transparent pricing, and India-specific infrastructure (Bangalore BLR1 region).

| Service | Provider | Purpose | Specifications |
|---|---|---|---|
| Droplets | DigitalOcean | Compute | 4 vCPU, 8GB RAM, 100GB SSD, Ubuntu 22.04 |
| Spaces | DigitalOcean | Object Storage | S3-compatible, CDN, BLR1 region, Private ACL |
| Load Balancer | DigitalOcean | Traffic Distribution | SSL termination, Health checks, WebSocket support |
| GradientAI | DigitalOcean AI | LLM Inference | Llama 3.3 70B, OpenAI-compatible API |
| Knowledge Bases | DigitalOcean AI | Vector Database | RAG, Embeddings, Document indexing |

## 4.6 Development & Deployment Tools

| Tool | Version | Purpose | Benefits |
|---|---|---|---|
| Docker | 24.0+ | Containerization | Consistency, Isolation, Easy deployment |
| Docker Compose | 2.x | Multi-container orchestration | Local development, Service dependencies |
| Air | Latest | Live reload (Go) | Fast feedback, Incremental builds |
| Turbopack | Integrated | Frontend bundler | 5x faster builds, HMR with state preservation |
| GitHub Actions | Latest | CI/CD | Automated testing, Continuous deployment |
| Git | 2.x | Version control | Collaboration, History, Branching |

## 4.7 System Architecture

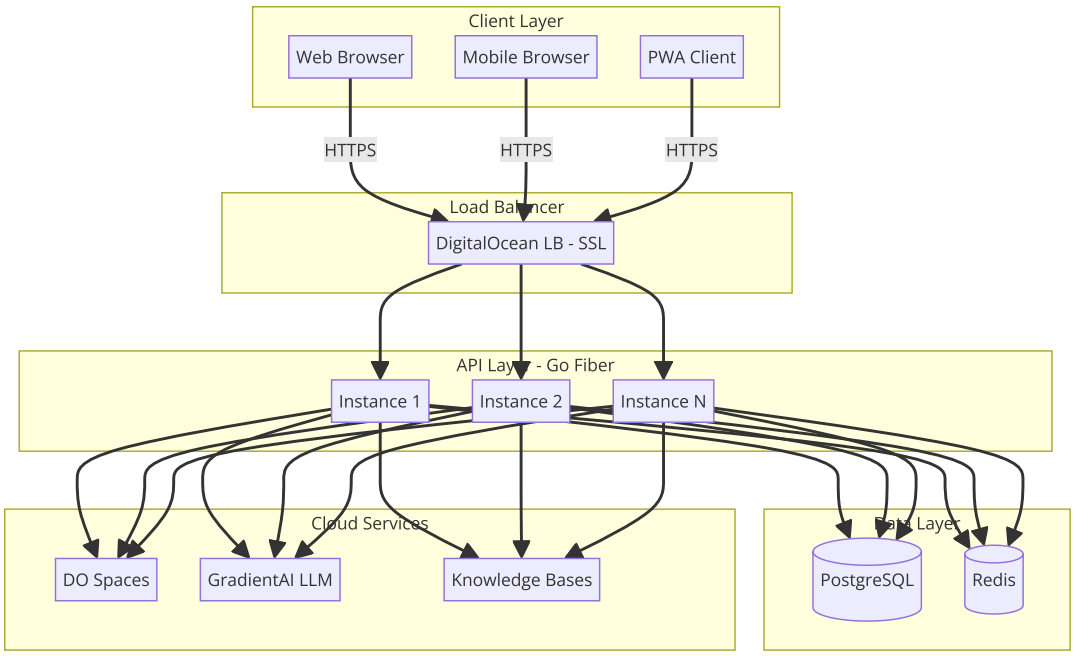



*Figure 4.1: System Architecture Overview*

## 4.8 Technology Selection Summary

| Decision | Choice | Rationale |
|---|---|---|
| Backend Language | Go over Node.js/Python | 3x more requests/sec than Node.js, single binary deployment, compile-time error checking |
| Database | PostgreSQL over MongoDB | Relational data model fits academic hierarchy, ACID guarantees, JSONB for flexibility |
| Frontend Framework | Next.js over CRA | SSR for SEO, 60% faster initial load, built-in routing and image optimization |
| Cloud Provider | DigitalOcean over AWS | Transparent pricing, Bangalore data center (15-30ms latency), integrated GradientAI |

# 5. SOFTWARE PROCESS MODEL

## 5.1 Methodology

The Study in Woods project follows an **Agile** software development methodology, implementing a hybrid approach combining Scrum for sprint management and Kanban for continuous feature flow. This methodology was chosen to enable rapid iteration and the ability to adapt to changing requirements during the academic project timeline.

Development spans 12 phases over 6 months (June - December 2024), with two-week sprints targeting 20-25 story points each. Phases overlap with continuous integration and testing running throughout.

## 5.2 Sprint Structure

| Activity | Frequency | Duration | Deliverables |
|---|---|---|---|
| Sprint Planning | Every 2 weeks | 2 hours | Sprint backlog, Story estimates |
| Sprint Review | Every 2 weeks | 1 hour | Working software demo |
| Backlog Refinement | Weekly | 1 hour | Refined user stories |

## 5.3 Development Phases

| Phase | Weeks | Key Deliverables | LOC |
|---|---|---|---|
| 1: Project Setup | 1-2 | Monorepo structure, Docker Compose, initial DB schema | ~3,500 |
| 2: Authentication | 3-4 | JWT auth, login/register, password reset | |
| 3: Academic Hierarchy | 5-6 | University/Course/Semester/Subject CRUD | |
| 4: Document Upload | 7-8 | DigitalOcean Spaces integration, multipart upload | ~4,200 |
| 5: AI Knowledge Base | 9-10 | GradientAI KB integration, document indexing | |
| 6: Syllabus Extraction | 11-12 | Llama 3.3 70B extraction, structured JSON output | |
| 7: Chat Sessions | 13-14 | Session CRUD, message history | ~3,800 |
| 8: AI Chat | 15-16 | SSE streaming, KB-context responses | |
| 9: Citations | 17-18 | Source document references in responses | |
| 10: PYQ Extraction | 19-20 | Question extraction from exam papers | ~3,000 |
| 11: Analytics | 21-22 | Usage tracking, dashboard charts | |
| 12: Admin Panel | 23-24 | User management, system settings, deployment | |

## 5.4 CI/CD Pipeline

Automated via GitHub Actions on every push:

- **Linting:** golangci-lint (Go), ESLint (TypeScript)
- **Testing:** Unit tests (70% coverage required), integration tests via Docker Compose

- **Build:** Multi-stage Docker images
- **Deploy:** Zero-downtime deployment to DigitalOcean Droplet on main branch merge

## 5.5 Version Control

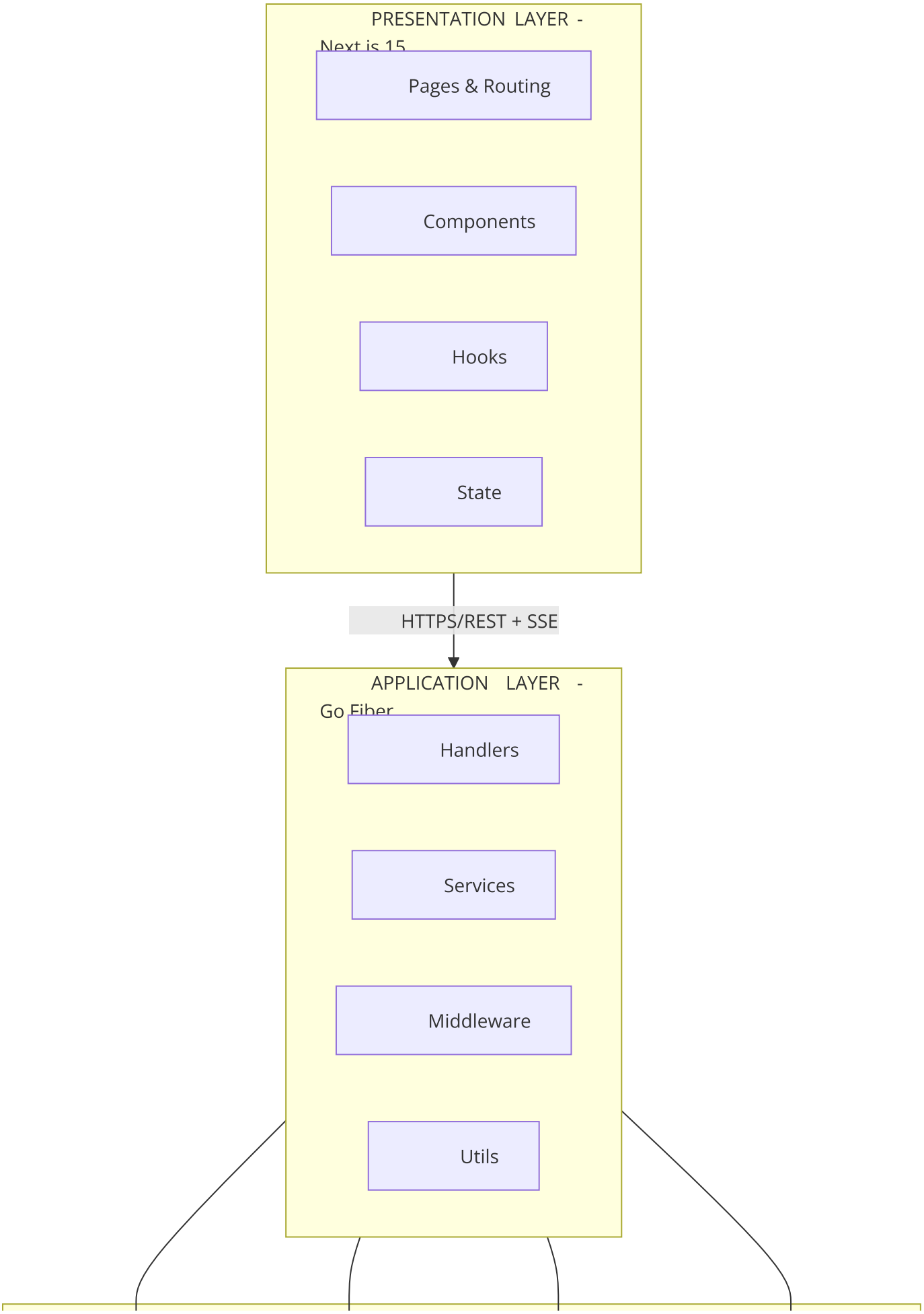
Git Flow branching with Conventional Commits:

- **main:** Production-ready code
- **develop:** Integration branch
- **feature/*:** Individual features
- Semantic versioning (MAJOR.MINOR.PATCH) for releases

# 6. DESIGN

## 6.1 System Architecture

Three-tier architecture: Presentation (Next.js), Application (Go Fiber API), Data (PostgreSQL, Redis, DigitalOcean Spaces). Communication via RESTful APIs, SSE for streaming, S3 for storage.


PRESENTATION LAYER - Next.js 15
Pages & Routing
Components
Hooks
State
HTTPS/REST + SSE
APPLICATION LAYER - Go Fiber
Handlers
Services
Middleware
Utils

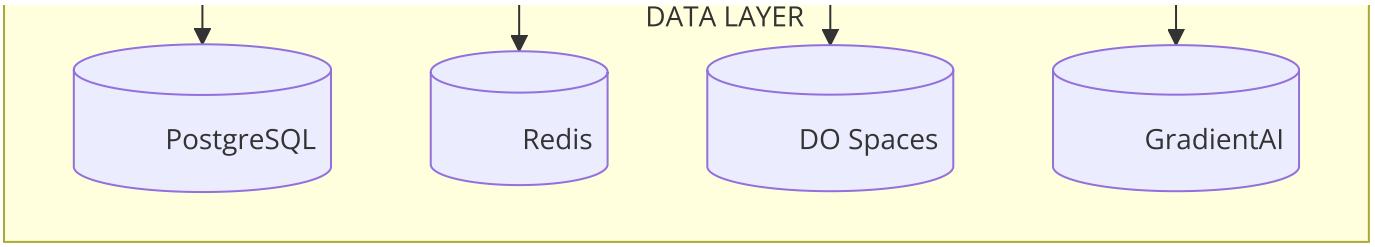



*Figure 6.1: Three-tier System Architecture*

## 6.2 Design Patterns Used

| Pattern | Description |
|---|---|
| **Repository Pattern** | GORM-based data access layer abstracts database operations |
| **Service Layer** | Business logic separated from handlers and data access |
| **Middleware Chain** | JWT auth, CORS, rate limiting, logging as composable middleware |
| **Provider Pattern** | React context providers for auth, theme, and query state |
| **Observer Pattern** | SSE for real-time streaming of AI responses and job status |
| **Factory Pattern** | Service initialization with dependency injection |

## 6.3 Data Flow Summary

### *6.3.1 Context Diagram (Level 0)*

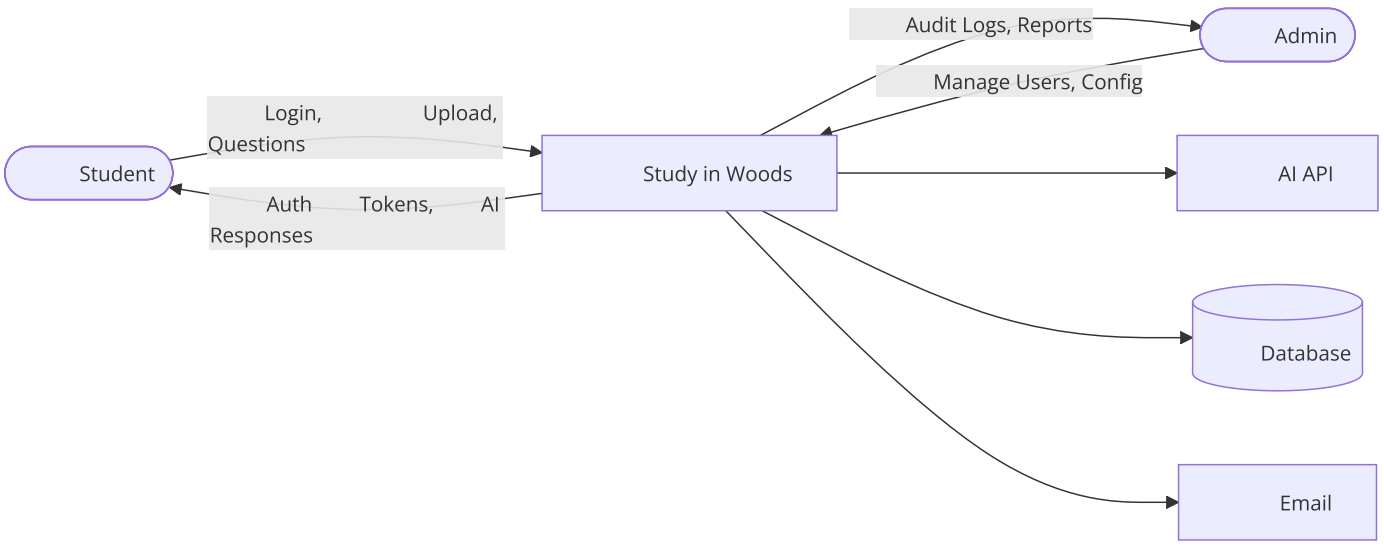



*Figure 6.2: Context Diagram (Level 0 DFD)*

### 6.3.2 System Decomposition (Level 1)

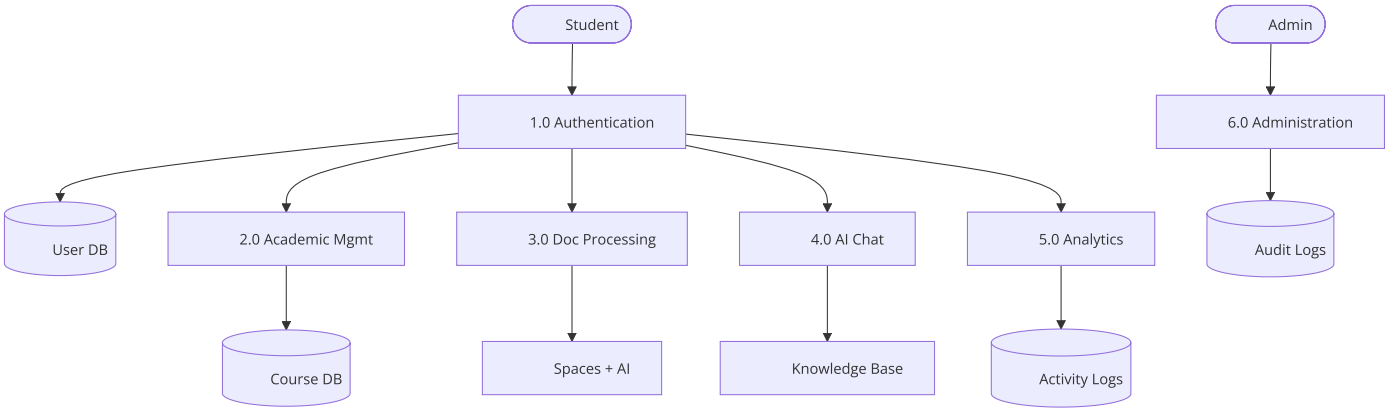
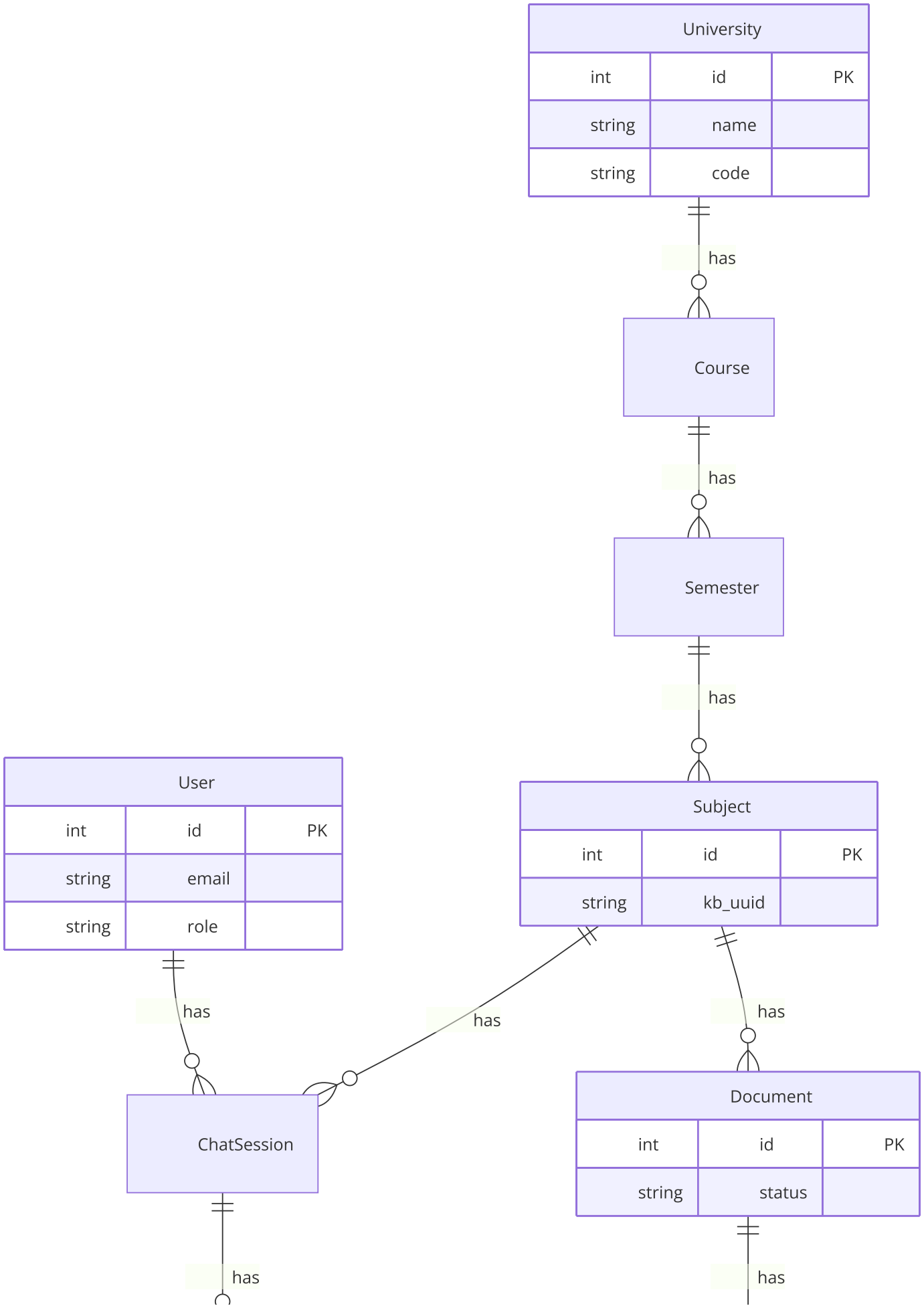



*Figure 6.3: System Decomposition (Level 1)*

## 6.4 Entity Relationship Diagram

Database: 30+ tables. Core hierarchy: University → Course → Semester → Subject → Documents/ChatSessions.


University
int id PK
string name
string code
has
Course
has
Semester
has
Subject
int id PK
string kb_uuid
User
int id PK
string email
string role
has
has
has
ChatSession
Document
int id PK
string status
has
has

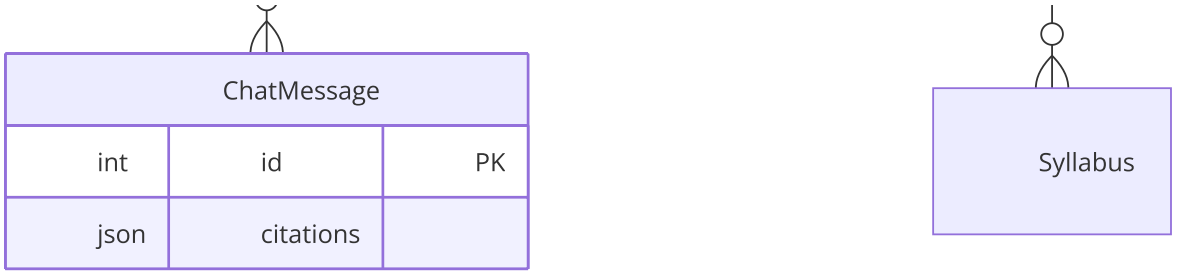



*Figure 6.4: Entity Relationship Diagram*

## 6.5 Key Sequence Flows

### 6.5.1 Authentication Flow

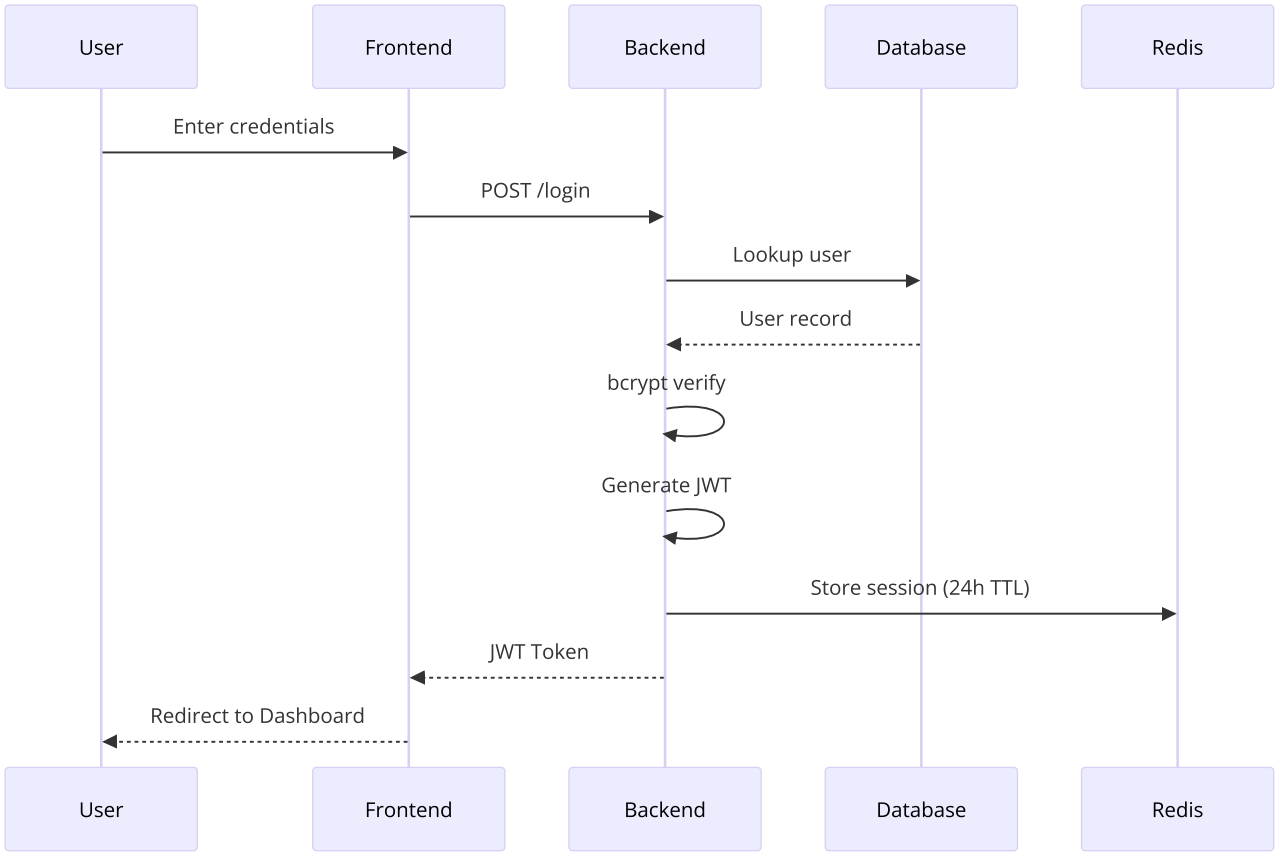



*Figure 6.5: Authentication Sequence Diagram*

### 6.5.2 Document Upload & AI Extraction

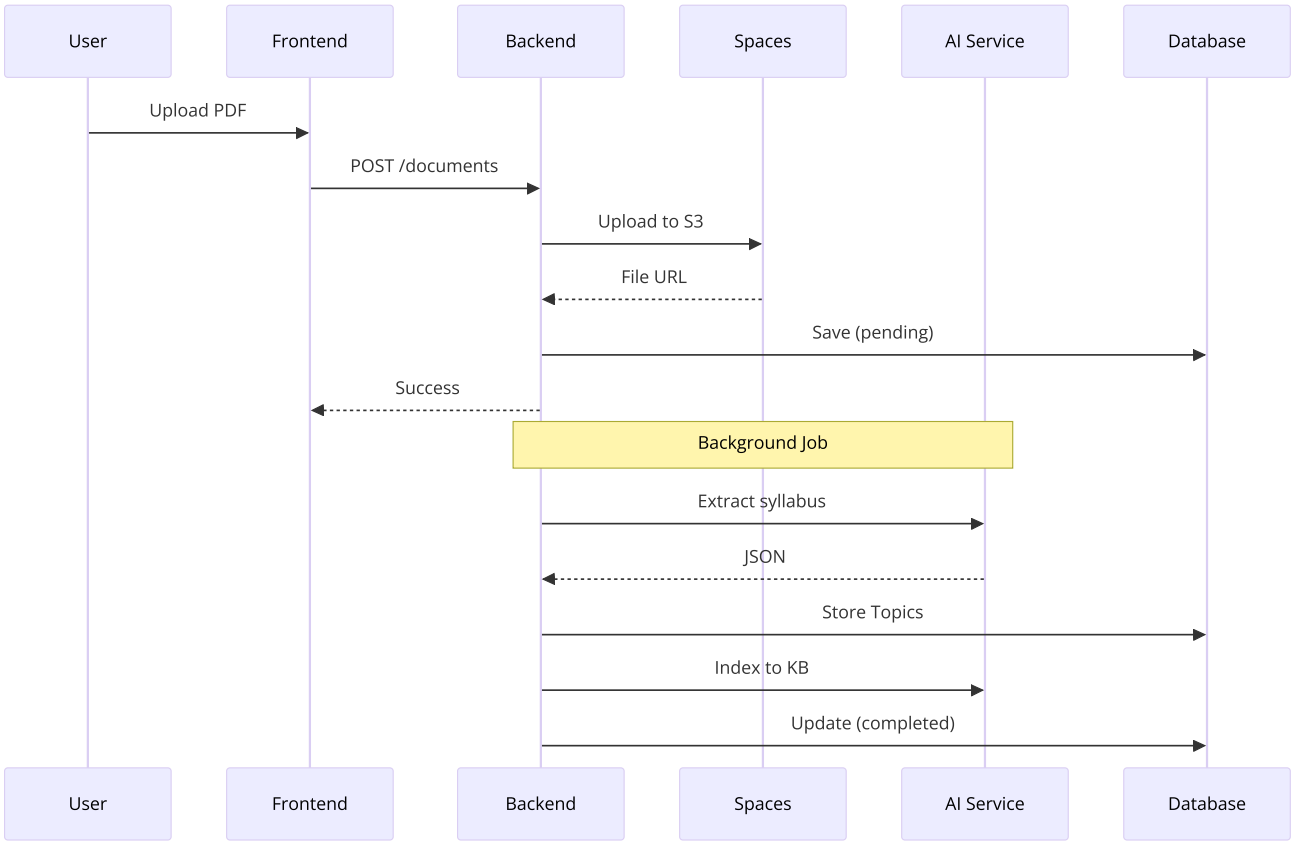



*Figure 6.6: Document Upload and AI Extraction Flow*

### 6.5.3 AI Chat with RAG

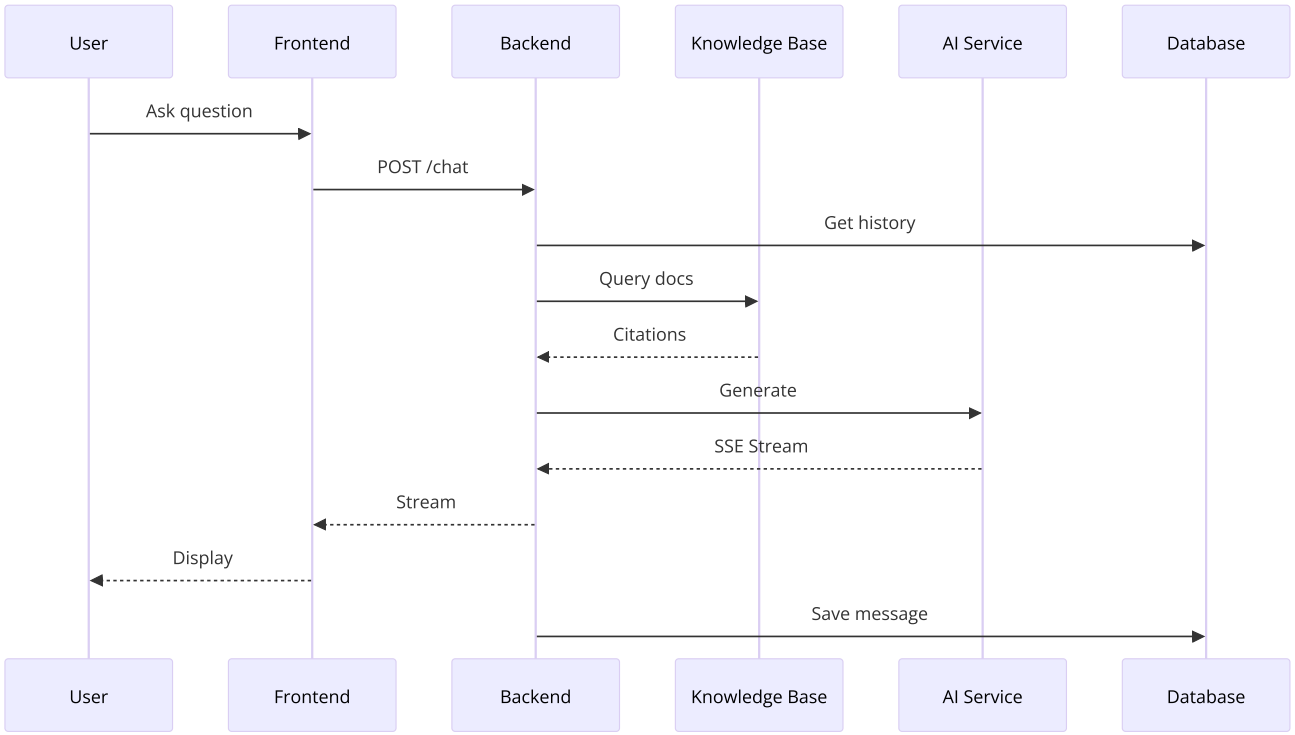



*Figure 6.7: AI Chat with RAG Flow*

## 6.6 Component Architecture

### 6.6.1 Backend Layers

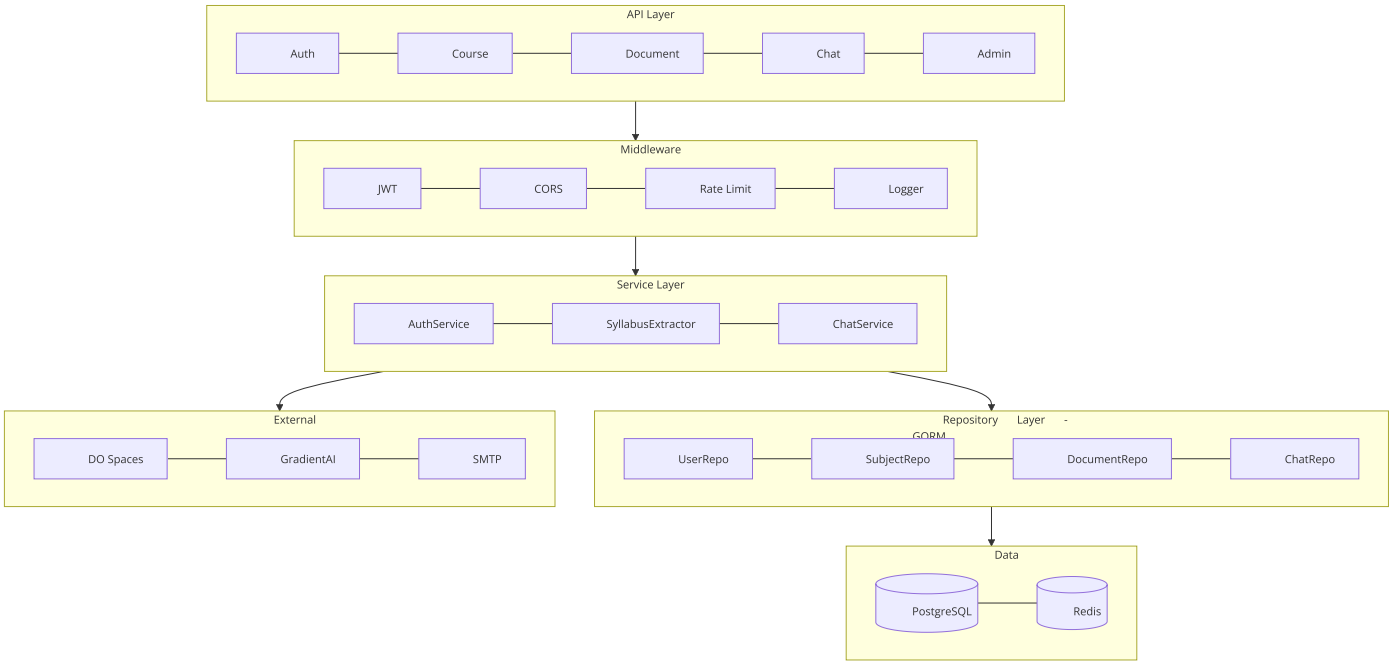



*Figure 6.8: Backend Layered Architecture*

### 6.6.2 Frontend Structure

The Next.js 15 frontend follows a modular architecture with clear separation of concerns:

| Layer | Components | Purpose |
|---|---|---|
| **App Router** | Next.js 15 App Directory | File-based routing with layouts |
| **Layout** | Header, Sidebar, Footer | Consistent UI shell across pages |
| **Auth Pages** | /login, /register | User authentication flows |
| **Dashboard** | Stats, Activity, Actions | Overview and quick actions |
| **Academic** | /universities, /courses, /subjects | Academic hierarchy management |
| **Chat** | /chat/[subjectId] | AI chat with sessions, messages, citations |
| **Analytics** | Charts, Stats, Activity Table | Usage insights and metrics |
| **Admin** | Users, Settings, Audit Logs | System administration |

| Layer | Components | Purpose |
| --- | --- | --- |
| **Providers** | Query, Theme, Auth | Global state and context |
| **Utils** | API Client (Axios), Hooks, Zod | Shared utilities and validation |

*Figure 6.9: Frontend Application Structure*

# 7. DATABASE

## 7.1 Database Overview

The Study in Woods platform uses PostgreSQL 15.x as its primary relational database management system, storing all persistent application data across **30+ tables**. The database schema follows normalization principles (3NF) to minimize data redundancy while maintaining referential integrity through foreign key constraints. GORM (Go Object Relational Mapping) library manages database operations, automatic migrations, and relationship handling.

## 7.2 Database Schema Summary

| Category | Tables | Purpose |
|---|---|---|
| User Management | users, jwt_token_blacklist | Authentication, authorization, session management |
| Academic Hierarchy | universities, courses, semesters, subjects | Educational structure and curriculum organization |
| Document Management | documents, syllabuses, syllabus_units, syllabus_topics | File storage, syllabus extraction, content indexing |
| Chat System | chat_sessions, chat_messages, chat_memories, chat_compacted_contexts | AI conversations, context management, memory optimization |
| PYQ System | pyq_papers, pyq_questions, pyq_question_choices, pyq_crawler_sources, pyq_crawled_papers | Previous year questions management and crawling |
| System & Audit | api_keys, api_key_usage_logs, user_activities, admin_audit_logs, app_settings | API management, activity tracking, configuration |
| Background Jobs | indexing_jobs, indexing_job_items, cron_job_logs | Asynchronous processing, job scheduling |
| User Engagement | user_notifications, user_courses | Notifications, enrollment tracking |

## 7.3 Core Table Definitions

### *7.3.1 users*

Primary table for user authentication and profile management.

| Column | Type | Key/Constraint |
|---|---|---|
| id | SERIAL | PRIMARY KEY |
| email | VARCHAR(255) | UNIQUE, NOT NULL |
| password_hash, password_salt | VARCHAR(255), BYTEA | NOT NULL (bcrypt) |
| name | VARCHAR(255) | NOT NULL |
| role | VARCHAR(20) | DEFAULT 'student' (student/admin) |
| semester, token_version | INTEGER | DEFAULT 1, 0 |
| created_at, updated_at, deleted_at | TIMESTAMP | Soft delete support |

**Relations:** Has many ChatSessions, ChatMessages, UserCourses, AdminAuditLogs

### *7.3.2 Academic Hierarchy (universities -> courses -> semesters -> subjects)*

| Table | Key Columns | Foreign Key |
|---|---|---|
| universities | id, name, code (UNIQUE), location, is_active | - |
| courses | id, name, code (UNIQUE), duration, description | university_id -> universities(id) |
| semesters | id, number, name | course_id -> courses(id) |
| subjects | id, name, code, credits, knowledge_base_uuid, agent_uuid | semester_id -> semesters(id) |

All tables include created_at, updated_at timestamps. CASCADE delete propagates through hierarchy.

### 7.3.3 documents

Stores uploaded PDFs and tracks indexing status with DigitalOcean Knowledge Base.

| Column | Type | Purpose |
|---|---|---|
| id | SERIAL | PRIMARY KEY |
| subject_id | INTEGER | FK -> subjects(id) |
| type | VARCHAR(20) | 'syllabus', 'pyq', 'book', 'reference', 'notes' |
| filename, file_size, page_count | VARCHAR, BIGINT, INT | File metadata |
| spaces_url, spaces_key | TEXT, VARCHAR | DigitalOcean Spaces storage |
| data_source_id, indexing_job_id | VARCHAR(100) | Knowledge Base integration |
| indexing_status | VARCHAR(20) | 'pending', 'in_progress', 'completed', 'failed' |

### 7.3.4 Syllabus Structure (syllabuses -> syllabus_units -> syllabus_topics)

| Table | Key Columns | Foreign Key |
|---|---|---|
| syllabuses | id, subject_name, subject_code, total_credits, extraction_status, raw_extraction | subject_id, document_id |
| syllabus_units | id, unit_number, title, description, hours | syllabus_id -> syllabuses(id) |
| syllabus_topics | id, topic_number, title, description, keywords | unit_id -> syllabus_units(id) |

### 7.3.5 Chat System

#### chat_sessions

| Column | Type | Key/Constraint |
|---|---|---|
| id | SERIAL | PRIMARY KEY |
| user_id | INTEGER | FK -> users(id) |
| subject_id | INTEGER | FK -> subjects(id) |
| title | VARCHAR(255) | Auto-generated or user-set |

### chat_messages

| Column | Type | Purpose |
|---|---|---|
| id, session_id, subject_id, user_id | INTEGER | PK and Foreign Keys |
| role | VARCHAR(20) | 'user', 'assistant', 'system' |
| content | TEXT | Message content |
| citations | JSONB | Knowledge Base citations array |
| tokens_used, model_used, response_time | INT, VARCHAR, INT | Usage analytics |
| is_streamed | BOOLEAN | SSE streaming flag |

**Additional Chat Tables:** chat_memories (conversation context), chat_memory_batches (batch processing), chat_compacted_contexts (compressed long-term memory)

### *7.3.6 System & Audit Tables*

| Table | Key Columns | Purpose |
|---|---|---|
| api_keys | id, user_id, key_hash, name, is_active | Encrypted API key storage |
| api_key_usage_logs | id, user_id, service, endpoint, status_code | API consumption tracking |
| user_activities | id, user_id, action, resource_type, resource_id, ip_address | User action tracking |
| admin_audit_logs | id, admin_id, action, target_type, target_id, changes (JSONB) | Admin action audit trail |
| app_settings | id, key (UNIQUE), value, description | Application configuration |
| jwt_token_blacklist | id, user_id, token_hash (UNIQUE), expires_at | Invalidated token tracking |

## 7.4 Entity Relationships

The database follows a hierarchical structure with cascading relationships:

**Primary Relationships:**

- universities (1) -> (M) courses -> (M) semesters -> (M) subjects
- subjects (1) -> (M) documents, syllabuses, chat_sessions
- users (1) -> (M) chat_sessions -> (M) chat_messages
- syllabuses (1) -> (M) syllabus_units -> (M) syllabus_topics
- users (1) -> (M) api_keys, user_activities, admin_audit_logs

**ER Diagram Reference:** The complete Entity-Relationship diagram is available in the Design section (Chapter 6), showing all 30+ tables with their relationships, cardinality, and key constraints.

## 7.5 Indexes and Performance

Strategic indexing optimizes query performance:

- **B-tree indexes:** All foreign keys (user_id, subject_id, session_id, course_id, semester_id)
- **Unique indexes:** Email addresses, codes, token hashes
- **Composite indexes:** (user_id, created_at) for user activity queries
- **Partial indexes:** indexing_status='pending' for background job processing
- **GIN indexes:** JSONB columns (citations, metadata) for @> containment queries

**Connection Management:** pgx driver maintains 25-100 concurrent connections with 1-hour max lifetime. Query optimization includes prepared statement caching, GORM preloading to prevent N+1 queries, and Redis caching with 5-minute TTL.

# 8. SCREENS

## 8.1 Authentication Screens

### *8.1.1 Login Screen*

**URL:** /login - Centered authentication form with email/password inputs and form validation.

**Components:** Email input, Password input, Remember Me checkbox, Sign In button, Forgot Password link

### *8.1.2 Registration Screen*

**URL:** /register - User registration with password strength indicator.

**Components:** Name input, Email input, Password input (with strength meter), Confirm Password, Terms checkbox

## 8.2 Dashboard & Navigation

### *8.2.1 Main Dashboard*

**URL:** /dashboard - Central hub with statistics, recent activity, and quick actions.

**Layout:** Sidebar (left, 250px), Header (top, with profile dropdown), Content area (main)

**Components:** Stats cards (4), Recent activity list, Quick action buttons, Navigation sidebar

## 8.3 Academic Management Screens

### *8.3.1 Universities List*

**URL:** /universities - Card layout of universities with search and filter.

**Features:** Search/filter, Card grid, Add/Edit/Delete (admin), View associated courses

#### *8.3.2 Course List*

**URL:** /courses?university_id=X - Courses organized by university with breadcrumbs.

**Features:** Breadcrumbs, Filter by university, Course cards, Semester navigation

#### *8.3.3 Subject Detail*

**URL:** /subjects/[id] - Comprehensive subject view with tabbed interface.

**Tabs:** Overview, Syllabus, Documents (with upload), Chat, Analytics

### 8.4 Document Management Screens

#### *8.4.1 Document Upload Interface*

**URL:** /subjects/[id]?tab=documents - Drag-and-drop upload with real-time progress.

**Components:** Dropzone, Upload queue (progress bars), Document list table, Filter/search

#### *8.4.2 Syllabus Viewer*

**URL:** /subjects/[id]?tab=syllabus - Hierarchical display of extracted syllabus.

**Features:** Accordion units, Expand/collapse all, Edit mode (admin), Export to PDF

## 8.5 Chat Interface

### *8.5.1 Chat Screen*

**URL:** /chat/[subjectId]?session=[sessionId] - AI-powered chat with document citations.

**Layout:** Sessions sidebar (250px), Chat area (flex), Citation panel (300px, collapsible)

**Components:** Session list, Message bubbles (markdown), Input box, Citation cards, Typing indicator

**Features:** Real-time streaming (SSE), Markdown rendering, Code syntax highlighting

### *8.5.2 Chat Session Management*

**Features:** Session list, Search sessions, New session button, Delete with confirmation

## 8.6 Analytics Dashboard

### *8.6.1 Analytics Overview*

**URL:** /analytics - Usage metrics and trends visualization.

**Charts:** Daily active users (line), Documents uploaded (bar), Chat by subject (pie), API calls (area)

**Components:** Stats cards, Charts (Recharts), Leaderboard table, Date range picker, Export CSV

## 8.7 Admin Panel

### *8.7.1 User Management*

**URL:** /admin/users - User administration with role management.

**Features:** User table (sortable), Search/filter, Edit role, Reset password, Delete user, Bulk actions

### *8.7.2 System Settings*

**URL:** /admin/settings - System-wide configuration interface.

**Sections:** General, Authentication, File Upload, AI Config, Rate Limits, Email, Advanced

### *8.7.3 Audit Logs*

**URL:** /admin/audit-logs - Read-only administrative action logs.

**Features:** Filterable table, Date range filter, Action type filter, JSON diff viewer, Export CSV

## 8.8 Responsive Design

All screens support three breakpoints: mobile (320px-767px), tablet (768px-1023px), and desktop (1024px+). Mobile collapses sidebar to hamburger menu and stacks content vertically.

# 9. TESTING

## 9.1 Testing Strategy

The project implements comprehensive testing through automated CI/CD pipeline, with all tests executing on every pull request. The testing pyramid prioritizes unit tests (70%), integration tests (20%), and end-to-end tests (10%).

## 9.2 Types of Testing

- **Unit Testing:** Go (testify) - 156 tests covering services, handlers, utilities; React (Jest/RTL) - 89 component and hook tests
- **Integration Testing:** API tests with real PostgreSQL/Redis via Docker; database relationship and constraint verification
- **End-to-End Testing:** Playwright browser automation for critical user journeys
- **Security Testing:** Authentication/authorization verification, input validation (SQL injection, XSS, path traversal), dependency scanning
- **Performance Testing:** Load testing with k6 (100-1000 concurrent users), API response time benchmarks, database query optimization

## 9.3 Test Cases Summary

| Test Suite | Cases | Coverage Area |
|---|---|---|
| Authentication | 12 | Register, Login, JWT, Password reset |
| Academic Hierarchy | 23 | Universities, Courses, Subjects, Relationships |
| Documents & Syllabus | 28 | Upload, Validation, Extraction, Storage |
| Chat & AI | 14 | Sessions, Messages, Streaming, Citations |
| Admin | 9 | User management, Settings, Audit logs |

## 9.4 E2E User Journeys

| User Journey | Steps | Duration | Pass Rate |
|---|---|---|---|
| User Registration & Login | 7 | 15s | 98% |
| Course Enrollment | 10 | 22s | 96% |
| Document Upload & Processing | 12 | 45s | 94% |
| AI Chat Interaction | 8 | 18s | 97% |
| Admin User Management | 9 | 20s | 99% |

## 9.5 Performance Benchmarks

| Test Type | Virtual Users | Duration | Result |
|---|---|---|---|
| Baseline Load | 100 | 5 min | 95% requests < 2s |
| Spike Test | 50 -> 500 | 10 min | No crashes |
| Soak Test | 200 | 30 min | Stable memory |

## 9.6 Test Results Summary

| Metric | Target | Current | Status |
|---|---|---|---|
| Unit Test Coverage | 70% | 76% | **Pass** |
| Integration Test Coverage | 60% | 68% | **Pass** |
| E2E Coverage (Critical Paths) | 100% | 100% | **Pass** |
| API Response Time (p95) | < 2s | 1.2s | **Pass** |
| Security Vulnerabilities (High/Critical) | 0 | 0 | **Pass** |
| CI Pipeline Duration | < 15 min | 10 min | **Pass** |
| Overall Test Pass Rate | > 95% | 97.8% | **Pass** |

# 10. BIBLIOGRAPHY

## 10.1 Core Technologies

[1] The Go Programming Language. Google, 2024. https://go.dev/doc/

[2] Fiber Web Framework v2. Fiber, 2024. https://docs.gofiber.io/

[3] Next.js 15 Documentation. Vercel, 2024. https://nextjs.org/docs

[4] React Documentation. Meta, 2024. https://react.dev/

[5] TypeScript Documentation. Microsoft, 2024. https://www.typescriptlang.org/docs/

[6] Tailwind CSS v4.0. Tailwind Labs, 2024. https://tailwindcss.com/docs

## 10.2 Database & Infrastructure

[7] PostgreSQL 15 Documentation. PostgreSQL, 2024. https://www.postgresql.org/docs/15/

[8] Redis Documentation. Redis, 2024. https://redis.io/docs/

[9] GORM - ORM for Golang. GORM, 2024. https://gorm.io/docs/

[10] Docker Documentation. Docker, 2024. https://docs.docker.com/

## 10.3 Cloud & AI Services

[11] DigitalOcean API & Spaces Documentation. DigitalOcean, 2024. https://docs.digitalocean.com/

[12] Meta Llama 3.3 Model. Meta AI, 2024. https://ai.meta.com/llama/

## 10.4 Academic References

[13] Lewis, P., et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." NeurIPS, 2020.

[14] Vaswani, A., et al. "Attention Is All You Need." NeurIPS, 2017.

[15] Jones, M., et al. "JSON Web Token (JWT) - RFC 7519." IETF, 2015.

## 10.5 Books & Standards

[16] Donovan, A. & Kernighan, B. "The Go Programming Language." Addison-Wesley, 2015.

[17] Kleppmann, M. "Designing Data-Intensive Applications." O'Reilly, 2017.

[18] W3C. "Web Content Accessibility Guidelines (WCAG) 2.1." W3C, 2018.

[19] OWASP Foundation. "OWASP Top Ten 2021." OWASP, 2021. https://owasp.org/Top10/