

TABLE OF CONTENTS

S.No.	Particulars	Page
1.	INTRODUCTION	2
	- About Project, Purpose, Overview	
2.	PROJECT UNDERSTANDING DOCUMENT	6
	- Problem Statement, Proposed Solution, Scope	
3.	REQUIREMENTS	10
	- Functional & Non-Functional Requirements	
4.	TECHNOLOGY USED	19
	- Frontend, Backend, Database, Cloud Services	
5.	SOFTWARE PROCESS MODEL	23
	- Agile Methodology, Development Phases	
6.	DESIGN	30
	- System Architecture, DFD, ER Diagram, Sequence Diagrams	
7.	DATABASE	44
	- Schema Design, Table Specifications	
8.	SCREENS	49
	- User Interface Layouts	
9.	TESTING	60
	- Test Cases, Results, Coverage	
10.	BIBLIOGRAPHY	68

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. **Authentication:** Secure JWT-based 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.

1.8 Vision and Mission

Vision: To revolutionize academic learning through AI-powered intelligent systems that make quality education accessible, personalized, and efficient for every student regardless of their location or resources.

Mission: Study in Woods is committed to:

- **Simplify Academic Organization:** Provide a unified platform where students can access all their course materials, syllabi, and study resources in one place
- **Enable AI-Assisted Learning:** Deliver 24/7 intelligent study assistance through context-aware AI that understands course content and provides accurate, citation-backed answers
- **Automate Content Processing:** Eliminate manual effort in organizing syllabi by using advanced NLP to automatically extract and structure academic content
- **Support Data-Driven Study Planning:** Help students track their progress, identify knowledge gaps, and optimize their study time through analytics and insights
- **Bridge Understanding Gaps:** Connect syllabus topics with explanations, making complex subjects more accessible through conversational AI

1.9 Key Innovations

Study in Woods introduces several innovative features that distinguish it from traditional learning management systems:

Innovation	Description	Benefit
AI-Powered Syllabus Extraction	Uses Llama 3.3 70B LLM to parse PDF syllabi and extract structured units, topics, and learning outcomes automatically	Saves 2+ hours per subject; 85% extraction accuracy
Subject-Specific Knowledge Bases	Each subject gets its own vector knowledge base in DigitalOcean AI, enabling context-aware responses	AI answers are grounded in actual course materials
RAG-Based Chat System	Retrieval Augmented Generation ensures AI responses cite specific source documents	Verifiable answers with academic credibility
Real-Time Streaming Responses	Server-Sent Events (SSE) stream AI responses token-by-token	Responsive UX; no waiting for complete generation
Automatic Document Indexing	Uploaded documents are automatically indexed into subject knowledge bases	Zero manual configuration; instant searchability
Citation-Backed Answers	Every AI response includes citations linking to source documents and page numbers	Students can verify and explore further

1.10 Future Scope

The platform is designed with extensibility in mind. The following enhancements are planned for future releases:

Phase	Feature	Description	Timeline
Phase 2	Mobile Applications	Native iOS and Android apps with offline sync capabilities	Q1 2025
	Offline Mode	Download subjects for offline study with background sync when connected	Q1 2025
Phase 3	Video Lecture Processing	AI-powered transcription and summarization of video lectures	Q2 2025
	Multi-Language Support	Support for Hindi, regional languages, and translation features	Q2 2025
Phase 4	Collaborative Study Groups	Real-time collaboration features for group study sessions	Q3 2025
	LMS Integration	Integration with Moodle, Canvas, and other learning management systems	Q3 2025
Phase 5	Adaptive Learning Paths	AI-generated personalized study plans based on learning patterns	Q4 2025
	Assessment Generation	Automatic quiz and practice test generation from syllabus topics	Q4 2025

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
Authentication	Registration, login, 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	Academic content 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

3.6 Use Case Diagram

The following diagram illustrates the primary use cases and their relationships with system actors:

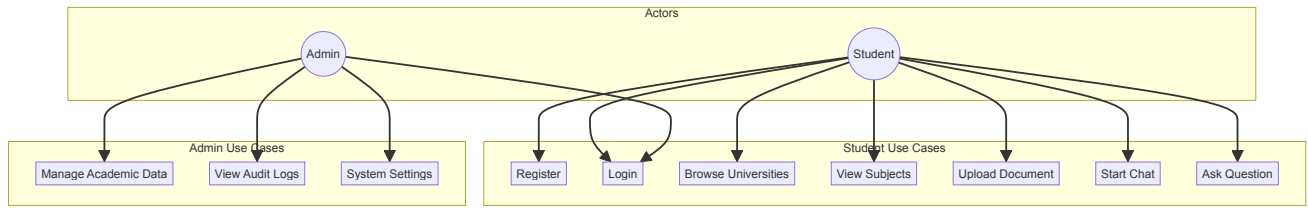


Figure 3.1: Use Case Diagram

3.7 Detailed Use Cases

3.7.1 UC-01: User Registration

Use Case ID	UC-01
Name	User Registration
Actor	Student (Primary)
Precondition	User has valid email address; email not already registered
Main Flow	<ol style="list-style-type: none">1. User navigates to registration page2. User enters name, email, password, and confirms password3. System validates input (email format, password strength)4. System checks email uniqueness5. System creates user account with hashed password6. System sends verification email7. User is redirected to login page with success message
Alternative Flow	<ol style="list-style-type: none">4a. Email already exists → Display error "Email already registered"3a. Weak password → Display password requirements
Postcondition	New user account created in database; verification email sent

3.7.2 UC-07: Upload Document

Use Case ID	UC-07
Name	Upload Document
Actor	Student (Primary), Admin
Precondition	User authenticated; subject selected; file is PDF format
Main Flow	<div>1. User navigates to subject's document tab</div> <div>2. User selects document type (syllabus, notes, PYQ, book)</div> <div>3. User drags/selects PDF file (max 10MB)</div> <div>4. System validates file type and size</div> <div>5. System uploads file to DigitalOcean Spaces</div> <div>6. System creates document record with "pending" status</div> <div>7. Background job indexes document to Knowledge Base</div> <div>8. If type=syllabus, extraction job is triggered</div> <div>9. Status updates to "completed" when processing finishes</div>
Alternative Flow	<div>4a. Invalid file type → Error "Only PDF files allowed"</div> <div>4b. File too large → Error "Maximum file size is 10MB"</div> <div>7a. Indexing fails → Status set to "failed", admin notified</div>
Postcondition	Document stored in Spaces; indexed in Knowledge Base; syllabus extracted if applicable

3.7.3 UC-11: Ask Question (AI Chat)

Use Case ID	UC-11
Name	Ask Question
Actor	Student (Primary)
Precondition	User authenticated; chat session active; subject has indexed documents
Main Flow	<div>1. User types question in chat input</div> <div>2. User clicks send or presses Enter</div> <div>3. System retrieves conversation history (last 10 messages)</div> <div>4. System queries Knowledge Base for relevant document chunks</div> <div>5. System constructs prompt with context and citations</div> <div>6. System calls Llama 3.3 70B API with streaming enabled</div> <div>7. Response tokens stream to frontend via SSE</div> <div>8. Frontend renders markdown response in real-time</div> <div>9. Citations panel updates with source references</div> <div>10. Message saved to database with citations JSON</div>
Alternative Flow	<div>4a. No relevant documents → AI responds with general knowledge + disclaimer</div> <div>6a. API timeout → Error displayed, retry option shown</div>
Postcondition	AI response displayed; message persisted; citations available

3.7.4 UC-13: Manage Academic Data (Admin)

Use Case ID	UC-13
Name	Manage Academic Data
Actor	Admin (Primary)
Precondition	Admin authenticated with admin role
Main Flow	<div>1. Admin navigates to admin panel</div> <div>2. Admin can add/edit/delete universities, courses, semesters, subjects</div> <div>3. Admin selects entity to manage</div> <div>4. Admin makes changes (create, update, or delete)</div> <div>5. System validates and saves changes</div> <div>6. Changes logged to audit_logs</div> <div>7. Confirmation message displayed</div>
Alternative Flow	<div>5a. Validation fails → Display error message</div> <div>4a. Delete with dependencies → Warning "This will delete associated data"</div>
Postcondition	Academic data updated; action logged in admin_audit_logs

3.8 User Stories

The following user stories define features from the end-user perspective with acceptance criteria:

ID	User Story	Acceptance Criteria	Priority
US-01	As a student , I want to upload my syllabus PDF so that topics are automatically extracted and organized	<ul style="list-style-type: none"> • PDF uploads within 30 seconds • Extraction completes within 2 minutes • Units and topics visible in syllabus tab • Extraction accuracy >= 85% 	High
US-02	As a student , I want to ask questions about my subject so that I get accurate, sourced answers	<ul style="list-style-type: none"> • AI responds within 5 seconds (first token) • Response includes relevant citations • Answers are contextually accurate • Can click citations to view source 	High
US-03	As a student , I want to browse universities and courses so that I can find my academic program	<ul style="list-style-type: none"> • Universities load within 1 second • Filter by name works correctly • Course hierarchy is clear • Can navigate to any subject in 3 clicks 	High
US-04	As a student , I want to view my chat history so that I can review previous conversations	<ul style="list-style-type: none"> • All past sessions listed • Can search within sessions • Messages load with pagination • Citations preserved in history 	Medium
US-05	As a student , I want to see my study analytics so that I can track my learning progress	<ul style="list-style-type: none"> • Dashboard shows activity stats • Charts display usage over time • Subject-wise breakdown available • Export data option 	Medium
US-06	As an admin , I want to manage academic content so that I can keep the platform organized	<ul style="list-style-type: none"> • Can add/edit universities and courses • Can manage semesters 	High

ID	User Story	Acceptance Criteria	Priority
		and subjects • Can upload/delete documents • All actions are audit-logged	
US-07	As an admin , I want to view system audit logs so that I can monitor administrative actions	• Logs show who, what, when • Can filter by action type • Can export to CSV • JSON diff for changes	Medium
US-08	As a student , I want to access the platform on mobile so that I can study on the go	• Responsive design works on all devices • Touch-friendly interface • Chat works on mobile • No horizontal scrolling required	High

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

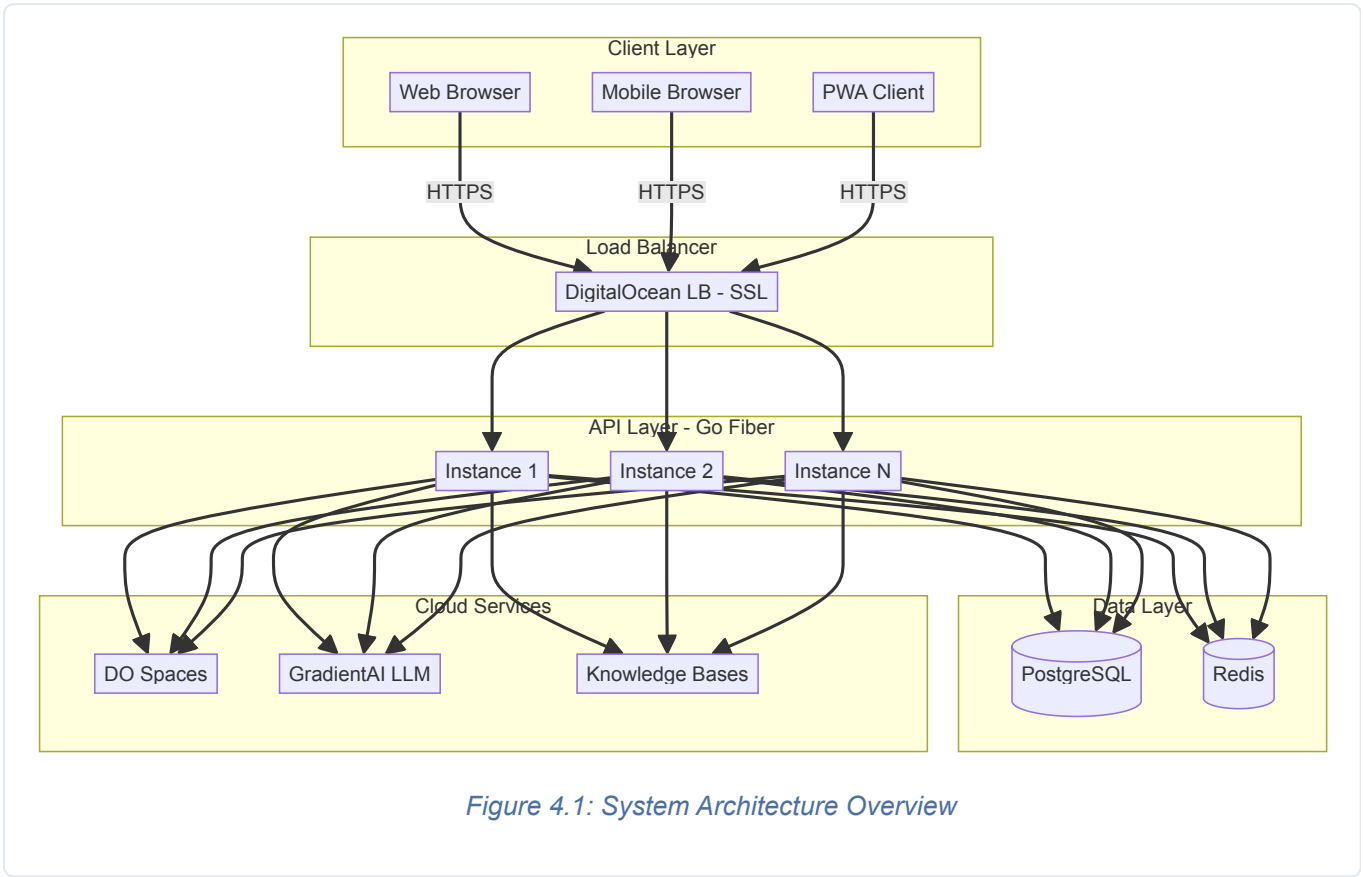
DigitalOcean provides the complete cloud infrastructure, selected for its simplicity, transparent pricing, and India-specific infrastructure (Bangalore BLR1 region).

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



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

5.6 Project Timeline (Gantt Chart)

The project spans 24 weeks (June - November 2024) with overlapping phases for continuous integration:

Phase	Jun	Jul	Aug	Sep	Oct	Nov
1. Project Setup	W1-2					
2. Authentication	W3-4					
3. Academic Hierarchy		W1-2				
4. Document Upload		W3-4				
5. AI Knowledge Base			W1-2			
6. Syllabus Extraction			W3-4			
7. Chat Sessions				W1-2		
8. AI Chat & Streaming				W3-4		
9. PYQ Extraction					W1-2	
10. Analytics Dashboard					W3-4	
11. Admin Panel						W1-2
12. Testing & Deployment						W3-4
Documentation	Continuous throughout project lifecycle					
Code Review & QA	Ongoing with each sprint					

Figure 5.1: Project Gantt Chart (June - November 2024)

5.7 Team Structure and Responsibilities

The development team consists of four members with clearly defined roles and responsibilities:

Team Member	Enrollment No.	Role	Primary Responsibilities
Sahil Chouksey	0208CA241050	Team Lead & Full Stack	<ul style="list-style-type: none">• Project architecture and technical decisions• AI/LLM integration (GradientAI, Knowledge Bases)• Code review and quality assurance• Deployment and DevOps setup• Sprint planning and task allocation
Ajay Kori	0208CA241009	Backend Developer	<ul style="list-style-type: none">• Go/Fiber API development• Database schema design (PostgreSQL)• Authentication and authorization (JWT)• Redis caching implementation• API documentation
Rohit Kumar Namdeo	0208CA241048	Frontend Developer	<ul style="list-style-type: none">• Next.js 15 / React 19 development• UI/UX implementation (Tailwind, shadcn/ui)• Responsive design and accessibility• State management (TanStack Query)• Frontend testing (Jest, RTL)
Vivek Raj Dhurwey	0208CA241071	QA & DevOps Engineer	<ul style="list-style-type: none">• Test case design and execution• CI/CD pipeline (GitHub Actions)• Docker containerization• Performance testing (k6)• Documentation and reporting

5.7.1 Communication and Collaboration

Activity	Frequency	Platform	Participants
Daily Standup	Daily (15 min)	Discord Voice	All team members
Sprint Planning	Bi-weekly (2 hrs)	Google Meet	All team members
Code Review	Per PR	GitHub	Lead + Developer
Sprint Retrospective	Bi-weekly (1 hr)	Google Meet	All team members
Documentation Sync	Weekly	Notion	All team members

5.8 Risk Management

The following risk assessment identifies potential project risks with mitigation strategies:

Risk ID	Risk Description	Likelihood	Impact	Risk Score	Mitigation Strategy
R-01	AI API Downtime: DigitalOcean GradientAI service unavailability	Medium	High	High	Implement graceful degradation; cache frequent responses; display user-friendly error messages; monitor API health
R-02	PDF Extraction Failures: Non-standard PDF formats causing extraction errors	Medium	Medium	Medium	Support multiple extraction strategies; provide manual override option; validate PDF before processing; log failures for analysis
R-03	Scope Creep: Feature requests exceeding project timeline	High	Medium	Medium	Strict sprint backlog management; prioritize MVP features; defer non-essential features to future phases; regular stakeholder communication
R-04	Performance Degradation: Slow response times under load	Low	High	Medium	Load testing with k6; database query optimization; Redis caching; horizontal scaling capability; CDN for static assets
R-05	Security Vulnerabilities: Data breaches or unauthorized access	Low	Critical	High	Security audits; dependency scanning; input validation; OWASP guidelines; encrypted storage; regular penetration testing
R-06	Team Member Unavailability: Illness or personal emergencies	Medium	Medium	Medium	Cross-training team members; documented codebase; pair programming; knowledge sharing sessions
R-07	Third-Party Dependency Issues: Breaking changes in libraries	Low	Medium	Low	Pin dependency versions; automated dependency updates with testing; maintain compatibility layer

5.9 Quality Assurance Process

Quality is ensured through a multi-layered approach integrated into the development workflow:

5.9.1 Code Quality Gates

Gate	Tool	Threshold	When
Linting	golangci-lint, ESLint	0 errors, 0 warnings	Pre-commit hook
Type Checking	TypeScript compiler	0 errors	Pre-commit hook
Unit Tests	Go test, Jest	70% coverage minimum	CI pipeline
Integration Tests	Docker Compose + API tests	All pass	CI pipeline
Security Scan	Dependabot, gosec	No high/critical vulnerabilities	Weekly + PR
Code Review	GitHub PR	1 approval required	Before merge

5.9.2 Definition of Done

A feature is considered "Done" when:

- Code is written and follows project coding standards
- Unit tests are written with $\geq 70\%$ coverage for new code
- Integration tests pass in CI environment
- Code review approved by at least one team member
- Documentation updated (API docs, README if applicable)
- No regression in existing functionality
- Feature deployed to staging and manually verified
- Product owner/stakeholder acceptance (for major features)

5.9.3 Deployment Checklist

#	Item	Verified By
1	All CI checks pass (lint, test, build)	Automated
2	Database migrations reviewed and tested	Backend Dev
3	Environment variables configured	DevOps
4	Docker images built and tagged	Automated
5	Staging deployment successful	Automated
6	Smoke tests pass on staging	QA
7	Production deployment (zero-downtime)	DevOps
8	Health checks passing	Automated
9	Monitoring alerts configured	DevOps
10	Rollback plan documented	Team Lead

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.

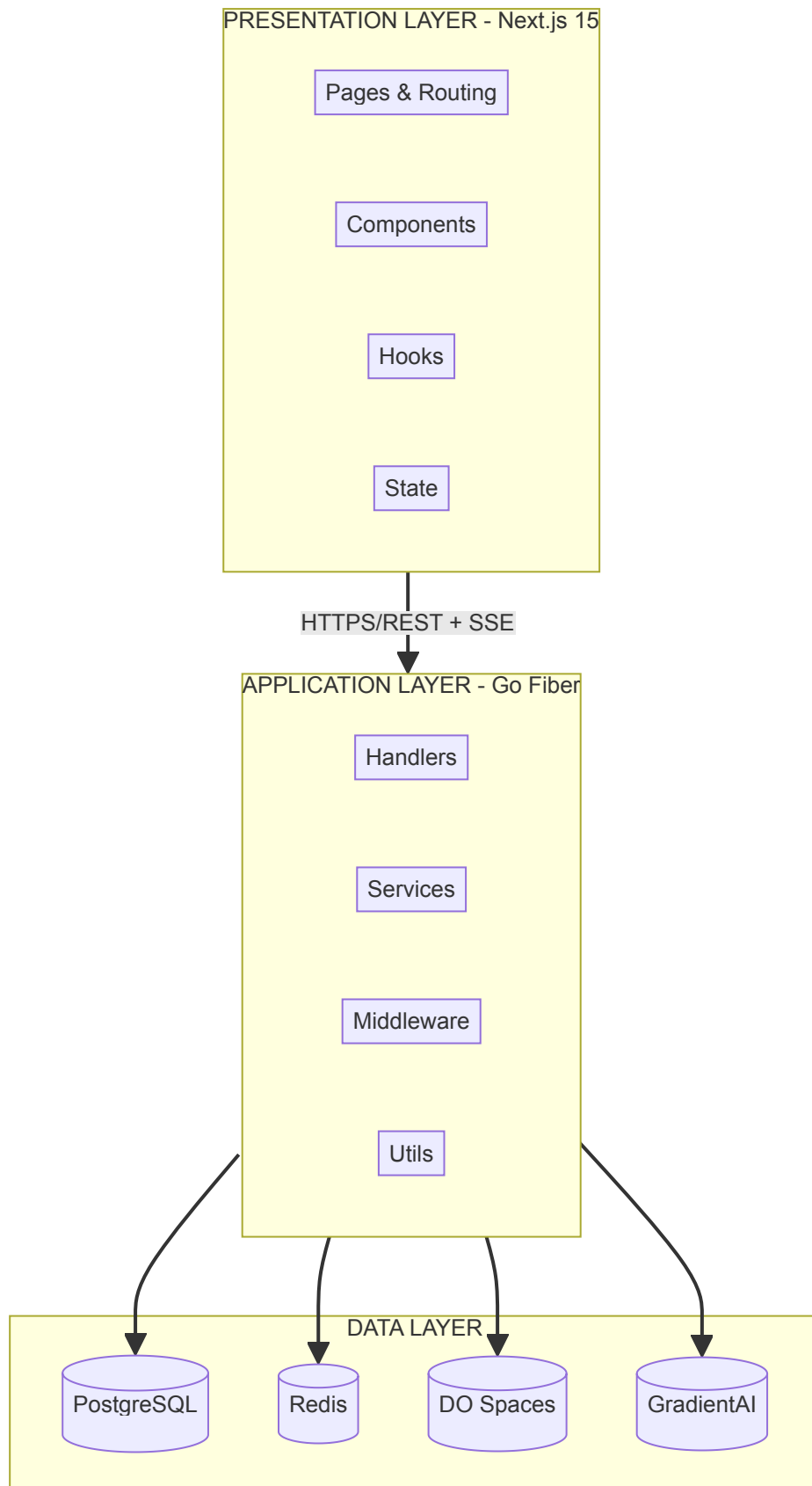


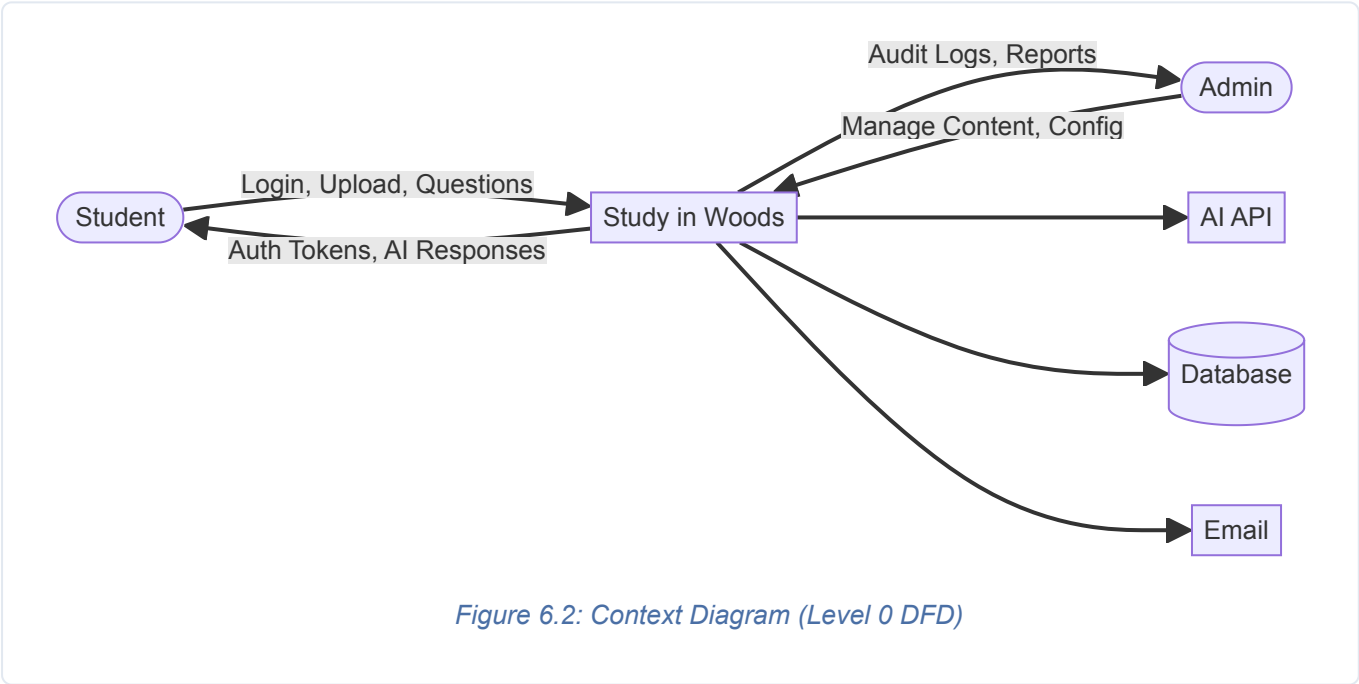
Figure 6.1: Three-tier System Architecture

6.2 Design Patterns Used

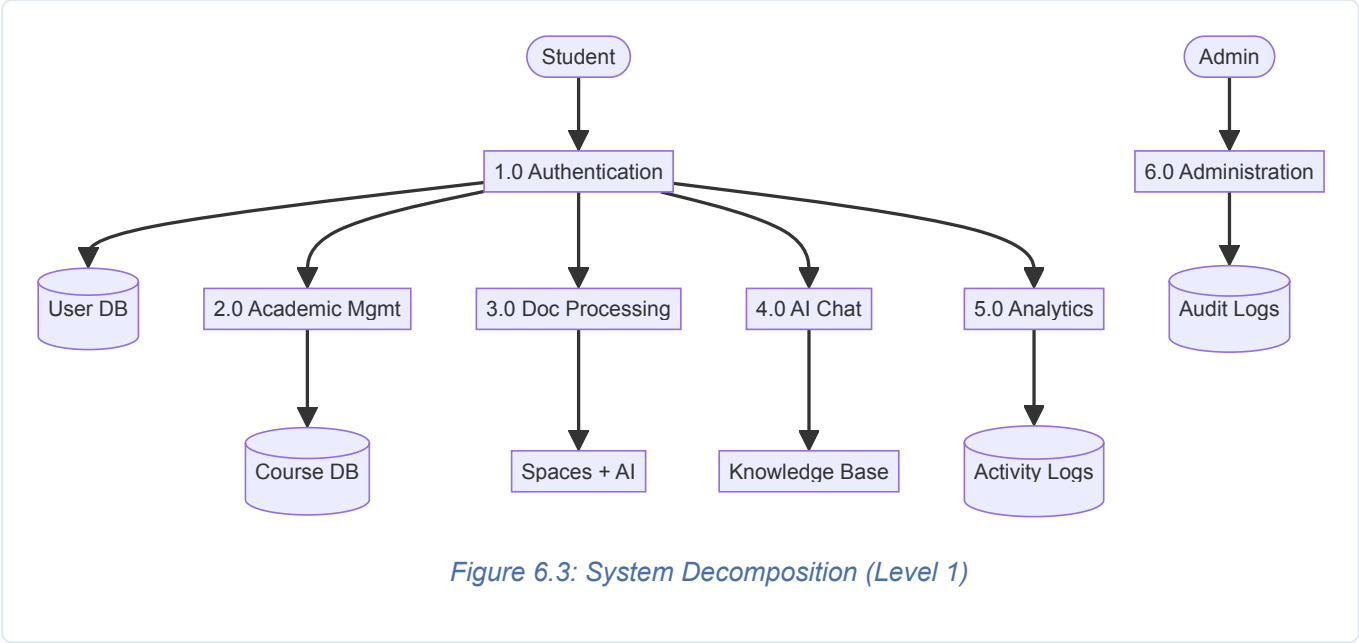
Pattern	Description
Repository Pattern	GORM-based data access layer abstracts database operations
Service Layer	Business logic separated from handlers and data access
Middleware Chain	JWT auth, CORS, rate limiting, logging as composable middleware
Provider Pattern	React context providers for auth, theme, 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)



6.3.2 System Decomposition (Level 1)



6.3.3 Document Processing Subsystem (Level 2)

The Document Processing subsystem (Process 3.0) handles file uploads, storage, and AI-powered content extraction:

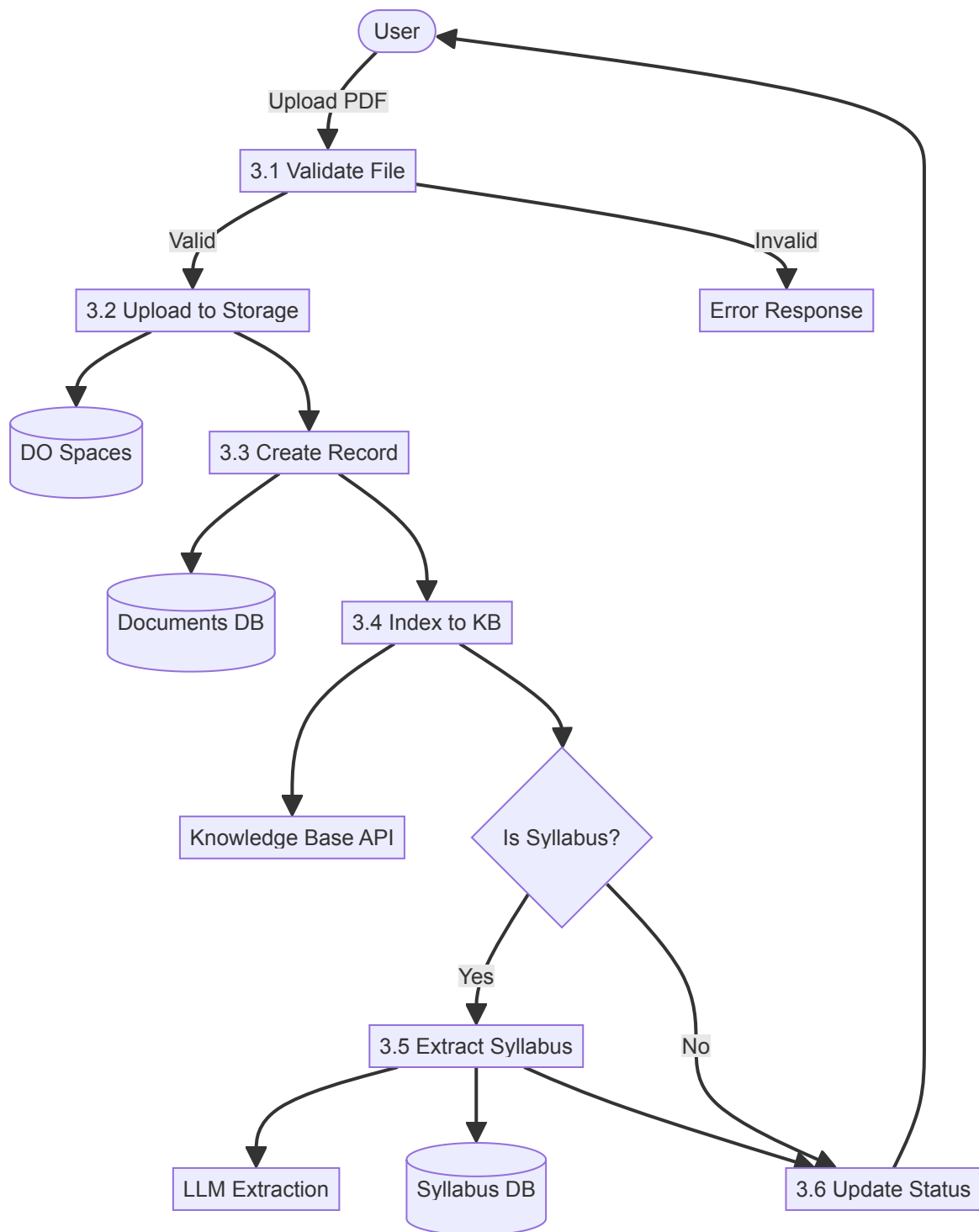


Figure 6.4: Document Processing Subsystem (Level 2 DFD)

6.3.4 AI Chat Subsystem (Level 2)

The AI Chat subsystem (Process 4.0) implements RAG-based conversational AI with citation support:

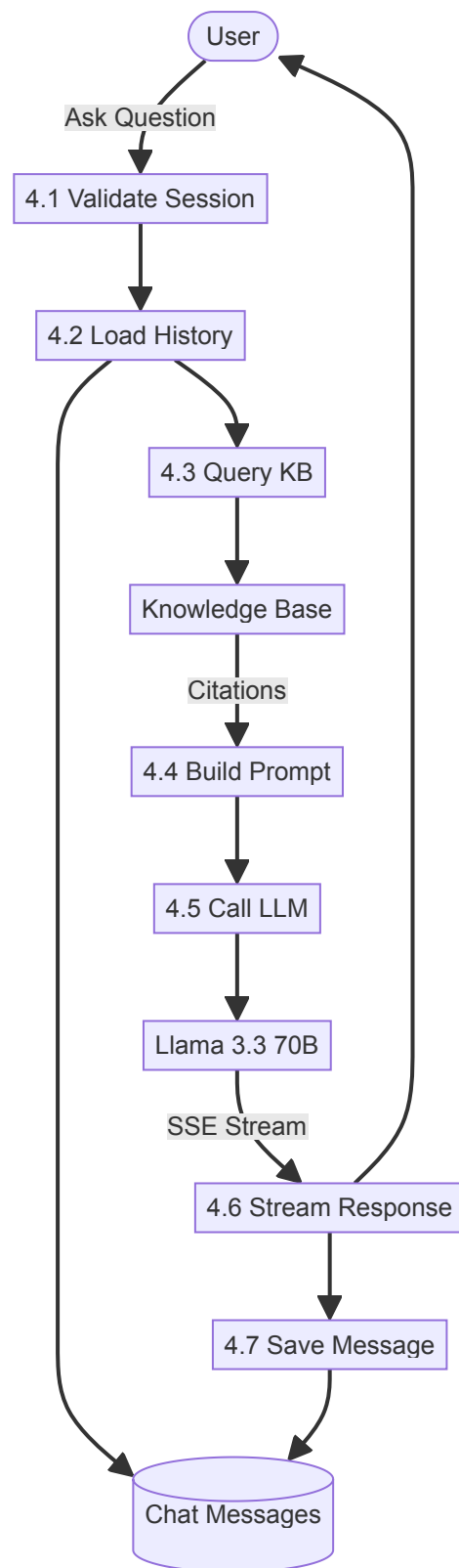


Figure 6.5: AI Chat Subsystem (Level 2 DFD)

6.4 Entity Relationship Diagram

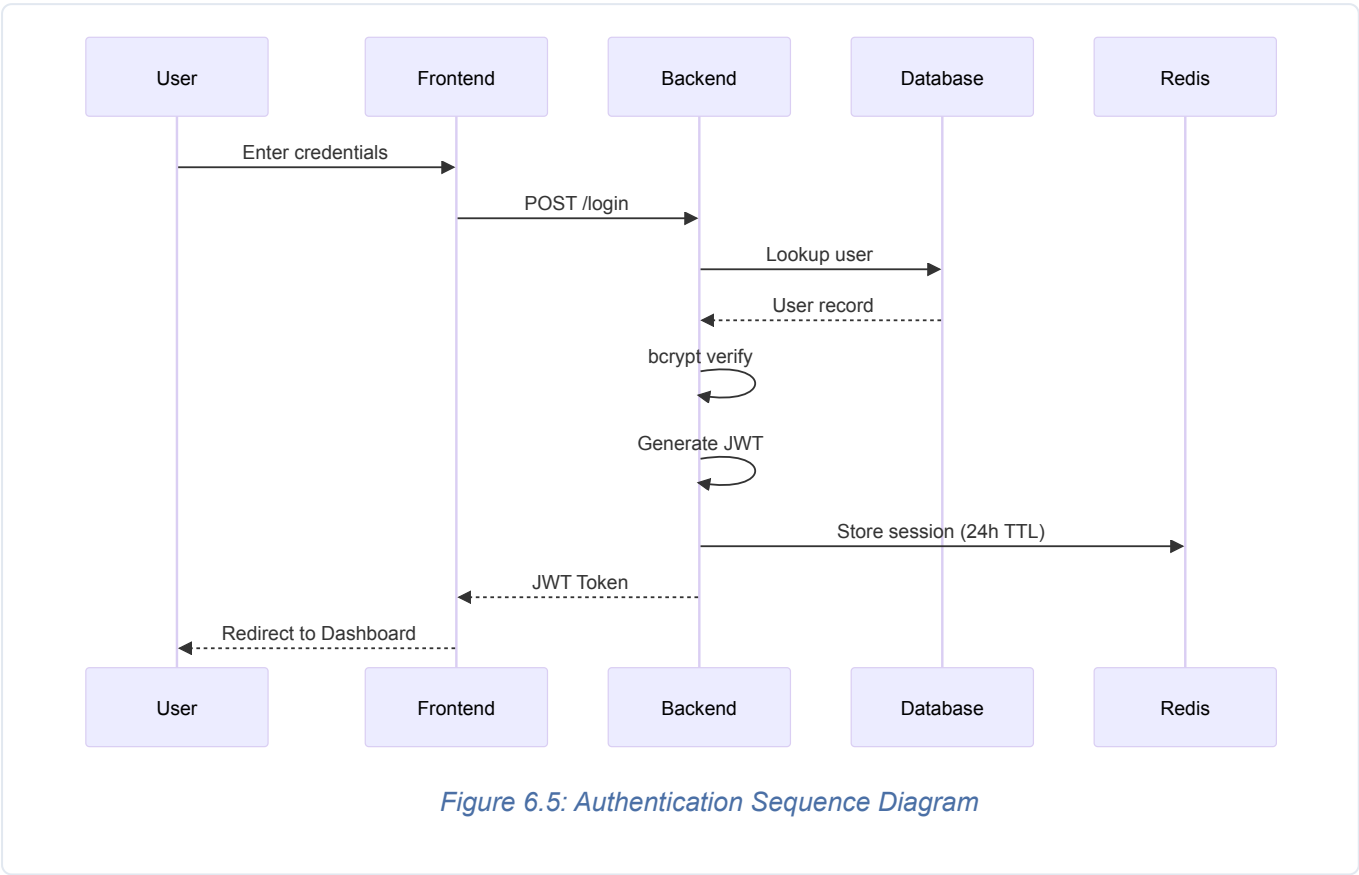
Database: 30+ tables. Core hierarchy: University → Course → Semester → Subject → Documents/ChatSessions.

Entity	Relationship	Related Entity	Cardinality
User	has	ChatSession	1:N
University	has	Course	1:N
Course	has	Semester	1:N
Semester	has	Subject	1:N
Subject	has	Document	1:N
Subject	has	ChatSession	1:N
Document	has	Syllabus	1:N
ChatSession	has	ChatMessage	1:N

Figure 6.4: Entity Relationships

6.5 Key Sequence Flows

6.5.1 Authentication Flow



6.5.2 Document Upload & AI Extraction

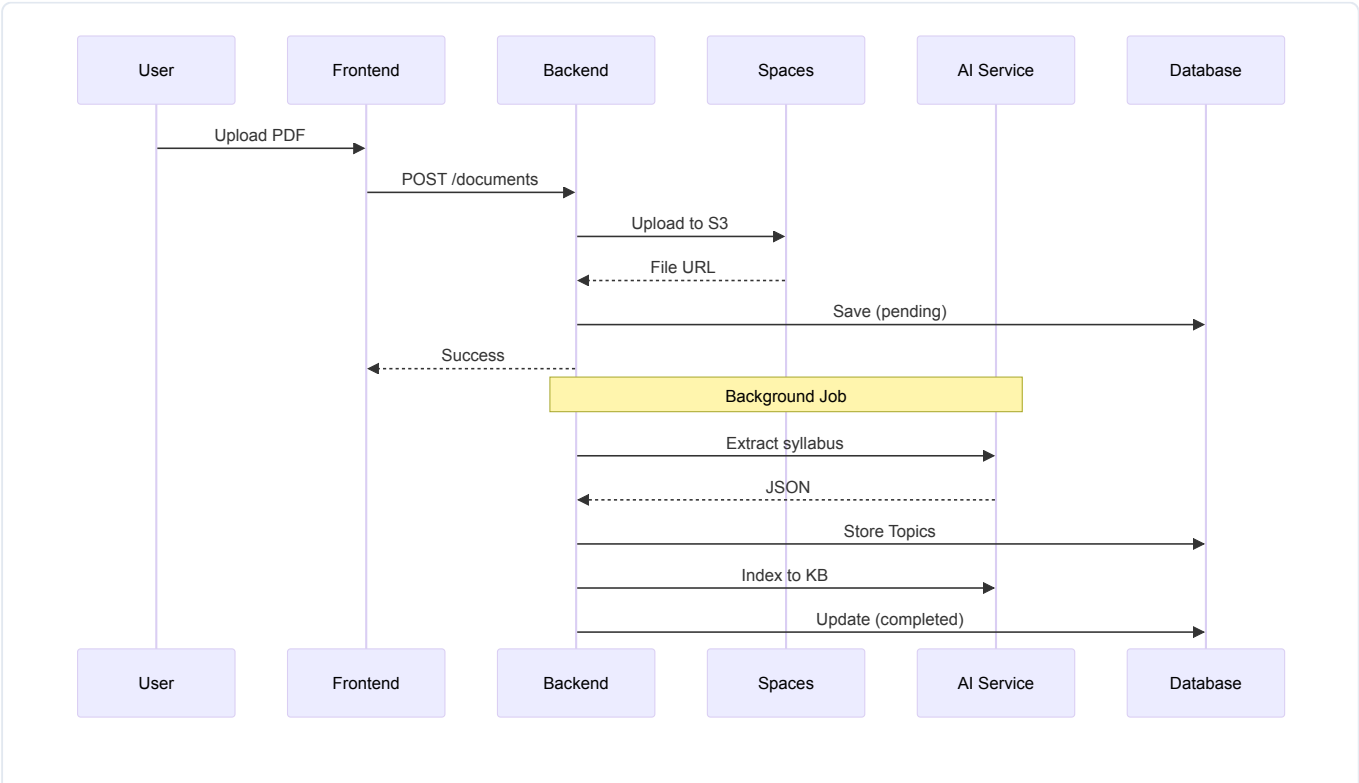


Figure 6.6: Document Upload and AI Extraction Flow

6.5.3 AI Chat with RAG

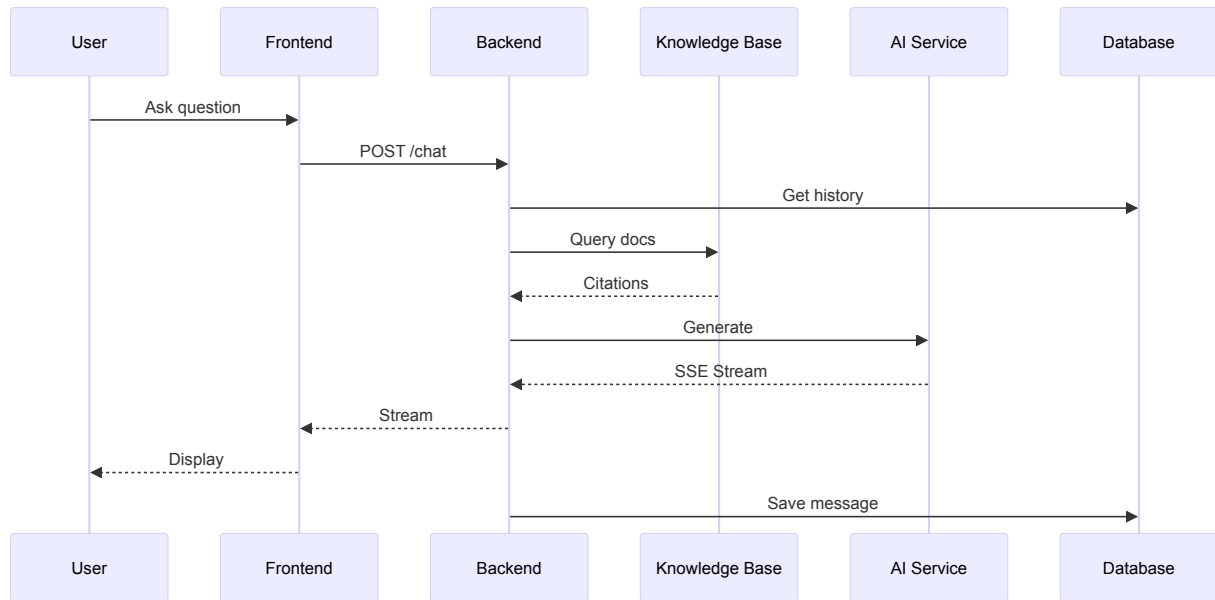
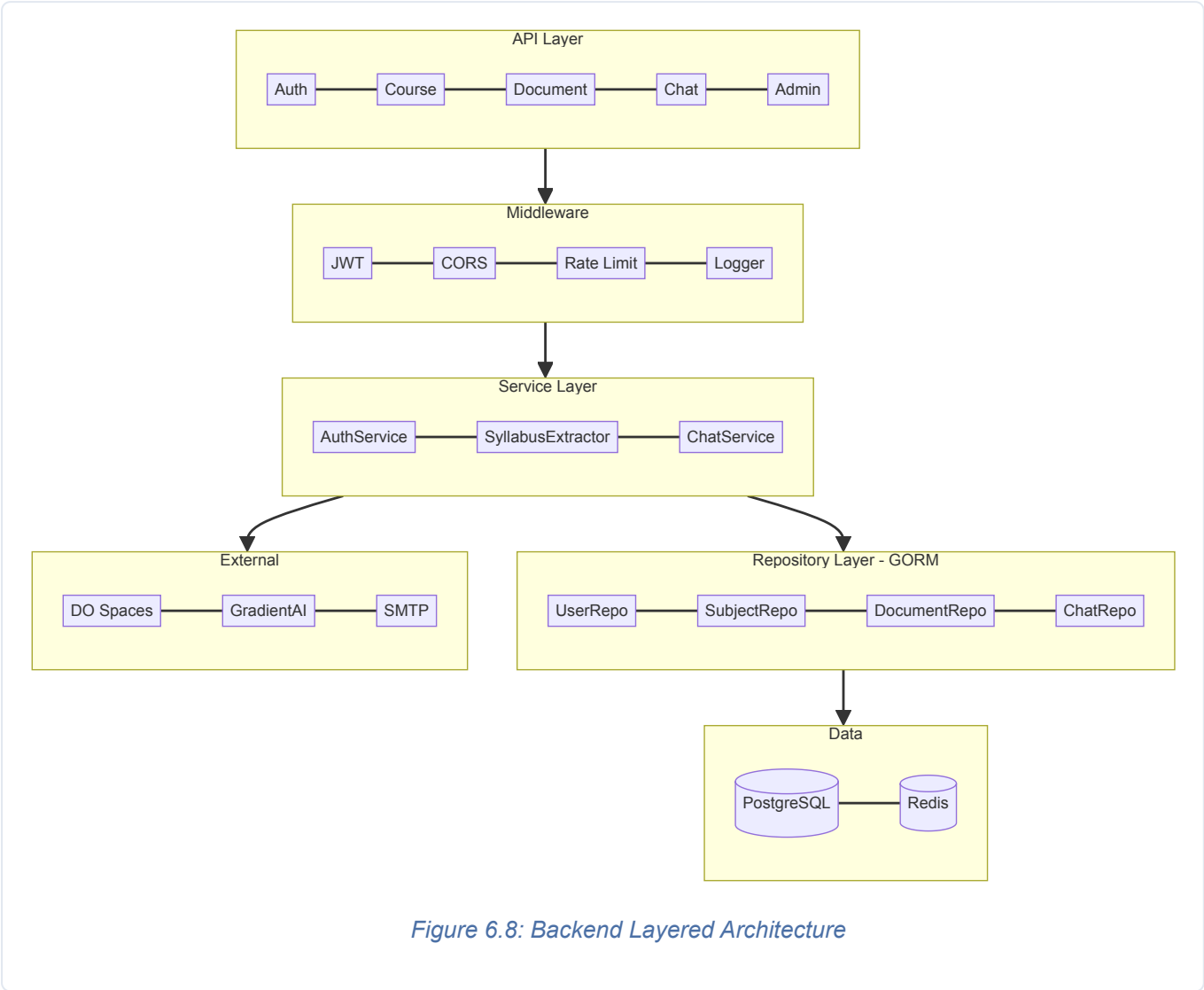


Figure 6.7: AI Chat with RAG Flow

6.6 Component Architecture

6.6.1 Backend Layers



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
Providers	Query, Theme, Auth	Global state and context
Utils	API Client (Axios), Hooks, Zod	Shared utilities and validation

Figure 6.9: Frontend Application Structure

6.7 API Endpoints Documentation

The backend exposes a RESTful API with 100+ endpoints. The following table documents the primary endpoints organized by module:

6.7.1 Authentication Endpoints

Method	Endpoint	Description	Auth	Request Body
POST	/api/v1/auth/register	Create new user account	No	name, email, password
POST	/api/v1/auth/login	Authenticate user, return JWT	No	email, password
POST	/api/v1/auth/logout	Invalidate current session	Yes	-
POST	/api/v1/auth/refresh	Refresh access token	Yes	refresh_token
POST	/api/v1/auth/forgot-password	Request password reset email	No	email
POST	/api/v1/auth/reset-password	Reset password with token	No	token, new_password
GET	/api/v1/auth/me	Get current user profile	Yes	-

6.7.2 Academic Management Endpoints

Method	Endpoint	Description	Auth
GET	/api/v1/universities	List all universities	Yes
POST	/api/v1/universities	Create university (Admin)	Admin
GET	/api/v1/universities/:id	Get university details	Yes
PUT	/api/v1/universities/:id	Update university (Admin)	Admin
DELETE	/api/v1/universities/:id	Delete university (Admin)	Admin
GET	/api/v1/courses	List courses (filter by university)	Yes
GET	/api/v1/semesters	List semesters (filter by course)	Yes
GET	/api/v1/subjects	List subjects (filter by semester)	Yes
GET	/api/v1/subjects/:id	Get subject with syllabus	Yes

6.7.3 Document & Syllabus Endpoints

Method	Endpoint	Description	Auth
GET	/api/v1/documents	List documents (filter by subject)	Yes
POST	/api/v1/documents	Upload document (multipart)	Yes
GET	/api/v1/documents/:id	Get document details	Yes
DELETE	/api/v1/documents/:id	Delete document	Yes
POST	/api/v1/documents/:id/reindex	Re-index document to KB	Yes
GET	/api/v1/syllabuses/:id	Get extracted syllabus	Yes
POST	/api/v1/syllabuses/:id/extract	Trigger syllabus extraction	Yes

6.7.4 Chat Endpoints

Method	Endpoint	Description	Auth
GET	/api/v1/chat/sessions	List user's chat sessions	Yes
POST	/api/v1/chat/sessions	Create new chat session	Yes
GET	/api/v1/chat/sessions/:id	Get session with messages	Yes
DELETE	/api/v1/chat/sessions/:id	Delete chat session	Yes
POST	/api/v1/chat/sessions/:id/messages	Send message, get AI response (SSE)	Yes
GET	/api/v1/chat/sessions/:id/messages	Get message history	Yes

6.7.5 Admin Endpoints

Method	Endpoint	Description	Auth
POST	/api/v1/admin/universities	Create university	Admin
PUT	/api/v1/admin/universities/:id	Update university	Admin
DELETE	/api/v1/admin/universities/:id	Delete university	Admin
POST	/api/v1/admin/courses	Create course	Admin
PUT	/api/v1/admin/courses/:id	Update course	Admin
DELETE	/api/v1/admin/courses/:id	Delete course	Admin
GET	/api/v1/admin/audit-logs	Get audit logs	Admin
GET	/api/v1/admin/settings	Get system settings	Admin
PUT	/api/v1/admin/settings	Update system settings	Admin
GET	/api/v1/admin/analytics	Get platform analytics	Admin

6.7.6 Sample API Request/Response

POST /api/v1/chat/sessions/:id/messages - Send message and receive AI response:

Request	Response (SSE Stream)
<pre>{ "content": "Explain polymorphism", "stream": true }</pre>	<pre>data: {"token": "Poly"} data: {"token": "morphism"} data: {"token": " is"} ... data: {"done": true, "citations": [{"source": "00P.pdf", "page": 42}]}</pre>

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

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
Layout	Centered card (400px width) on gradient background
Components	Logo, Email input, Password input (with show/hide toggle), Remember Me checkbox, Sign In button, "Forgot Password?" link, "Create Account" link
Validation	Email format validation, minimum password length (8 chars), real-time error display
States	Default, Loading (spinner on button), Error (red border + message), Success (redirect)
Accessibility	Form labels, aria-invalid on errors, focus management, keyboard navigation

8.1.2 Registration Screen

URL	/register
Layout	Centered card (450px width) with progress indicator
Components	Full Name input, Email input, Password input with strength meter (Weak/Medium/Strong), Confirm Password input, Terms & Conditions checkbox, Register button
Validation	Name required, email format + uniqueness check, password strength (uppercase, lowercase, number, 8+ chars), password match
States	Default, Validating (async email check), Error (field-level errors), Success (verification email sent modal)

8.2 Dashboard & Navigation

8.2.1 Main Dashboard Layout

URL	/dashboard
Layout Structure	Header (60px): Logo, Search bar, Notifications icon, Profile dropdown Sidebar (250px): Navigation menu, collapsed on mobile Main Content: Flex container with padding 24px
Stats Cards (4)	1. Total Subjects (with trend arrow) 2. Documents Uploaded (this month) 3. Chat Sessions (active/total) 4. Study Hours (this week)
Recent Activity	List showing last 10 actions: document uploads, chat messages, syllabus views with timestamps
Quick Actions	Buttons: "Upload Document", "Start Chat", "Browse Subjects", "View Analytics"

8.2.2 Navigation Sidebar

Section	Menu Items	Icon
Main	Dashboard, Universities, Subjects, Chat	Home, Building, Book, MessageSquare
Tools	Documents, Analytics, Calendar	File, BarChart, Calendar
Admin	Content, Settings, Audit Logs	Database, Settings, Shield
Footer	Help, Logout	HelpCircle, LogOut

8.3 Academic Management Screens

8.3.1 Universities List

URL	/universities
Layout	Page header + Search bar + Grid (3 columns desktop, 2 tablet, 1 mobile)
University Card	<ul style="list-style-type: none">• University logo/avatar• Name (bold, 16px)• Location (gray, 14px)• Course count badge• Status indicator (active/inactive)• Hover: elevation shadow + "View Courses" button
Admin Actions	Floating "Add University" button (bottom-right), Edit/Delete in card menu
Empty State	Illustration + "No universities found" + "Add University" CTA

8.3.2 Subject Detail Page

URL	/subjects/[id]
Header	Breadcrumb (University > Course > Semester > Subject), Subject name, Credits badge, Knowledge Base status
Tab Navigation	Overview Syllabus Documents Chat Analytics
Overview Tab	Subject description, Quick stats (documents, chat sessions, last activity), Recent documents list
Syllabus Tab	Accordion tree: Unit > Topics > Subtopics with expand/collapse all button
Documents Tab	Upload dropzone + Document table (name, type, size, status, date, actions)
Chat Tab	Embedded chat interface with session selector

8.4 Document Management

8.4.1 Document Upload Interface

Dropzone	Dashed border area (200px height), drag-over state (blue background), icon + "Drag PDF here or click to browse"
Upload Queue	Each item shows: <ul style="list-style-type: none">• File name (truncated)• File size• Progress bar (animated)• Status: Uploading → Processing → Indexing → Complete• Cancel button (during upload)
Document Table Columns	Checkbox Name Type (badge) Size Status Uploaded Actions (View, Download, Delete)
Type Badges	Syllabus (blue), Notes (green), PYQ (orange), Book (purple), Reference (gray)

8.5 Chat Interface

8.5.1 Chat Screen Layout

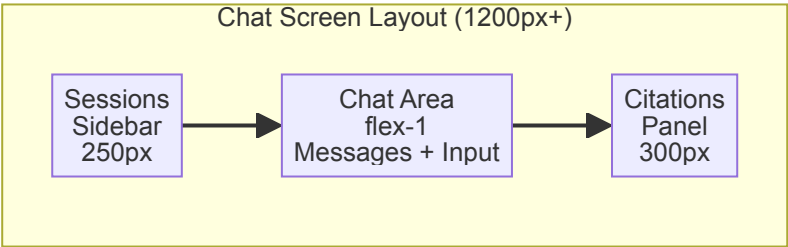


Figure 8.1: Chat Screen Three-Column Layout

Sessions Sidebar

- "New Chat" button (top)
- Search sessions input
- Session list: Title, Last message preview, Date
- Active session highlight (blue left border)
- Right-click menu: Rename, Delete

Chat Area

- Subject header with KB status indicator
- Message list (scrollable, auto-scroll to bottom)
- User messages: Right-aligned, blue background
- AI messages: Left-aligned, gray background, markdown rendered
- Typing indicator: Three animated dots during streaming
- Input: Textarea (auto-resize) + Send button

Citations Panel

- Collapsible (toggle button)
- Citation cards: Document name, Page number, Relevance score
- Click to open document in new tab
- "No citations" empty state

8.5.2 Message Components

Component	User Message	AI Message
Alignment	Right	Left
Background	#3182CE (blue)	#EDF2F7 (gray)
Text Color	White	#1A202C (dark)
Max Width	70%	80%
Features	Timestamp, Copy button	Markdown, Code blocks, Citation links, Copy, Regenerate

8.6 Analytics Dashboard

8.6.1 Analytics Overview

Date Range Picker	Presets: Today, Last 7 days, Last 30 days, Custom range
Stats Row	4 cards: Total Users Active Sessions Documents Processed AI Queries Each with: Value, % change from previous period, Trend arrow (green up / red down)
Charts	<ul style="list-style-type: none">Activity Over Time (Line chart - daily active users)Documents by Type (Pie chart)Chat Usage by Subject (Bar chart - top 10)API Response Times (Area chart)
Activity Table	Sortable table: User, Action, Resource, Timestamp with pagination (20 per page)

8.7 Admin Panel

8.7.1 Academic Content Management

URL	/admin/content
Tabs	Universities Courses Semesters Subjects
Content Table	Columns: Checkbox Name Code Status Created Actions Sortable by: Name, Code, Created Actions: Edit, Delete (with confirmation), View Details
Add/Edit Modal	Dynamic form based on entity type with validation
Bulk Actions	Delete selected, Export to CSV, Change status

8.7.2 System Settings

Section	Settings
General	Site name, Logo upload, Maintenance mode toggle
Authentication	Session timeout, Password requirements, 2FA toggle
File Upload	Max file size, Allowed types, Storage quota
AI Configuration	Model selection, Temperature, Max tokens, Rate limits
Email	SMTP settings, Email templates

8.8 User Flow Diagrams

8.8.1 Document Upload Flow

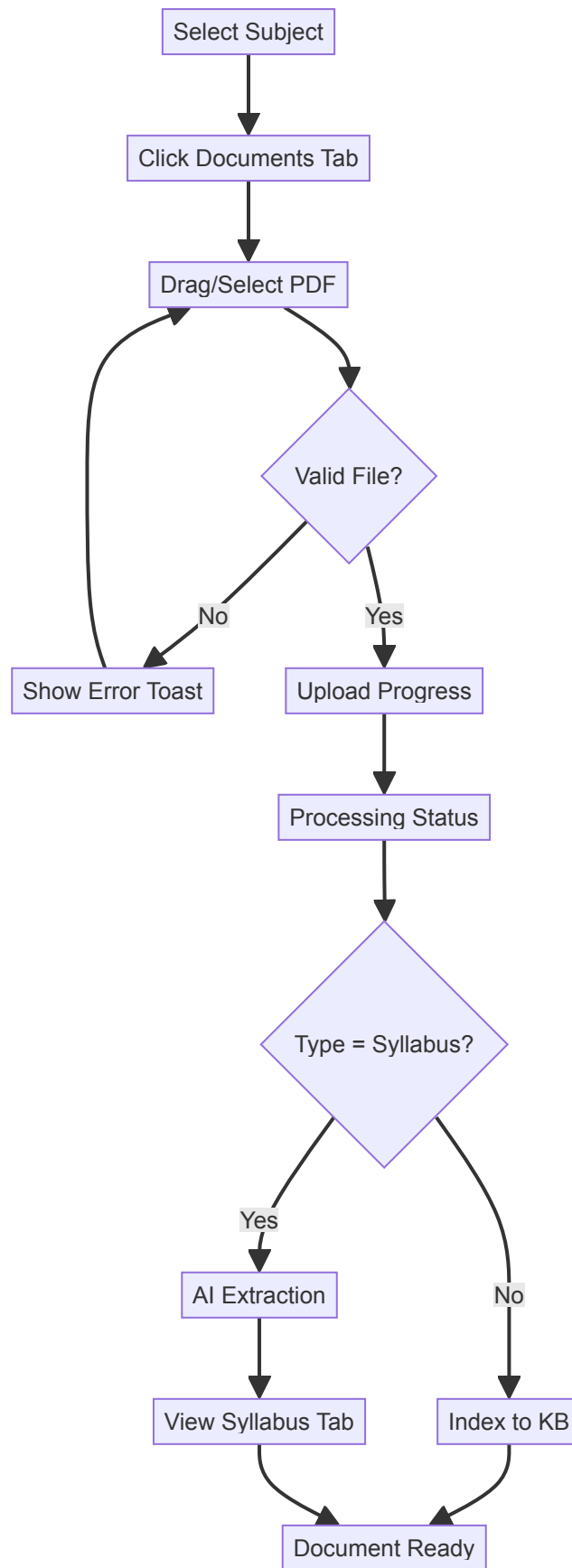


Figure 8.2: Document Upload User Flow

8.8.2 AI Chat Flow

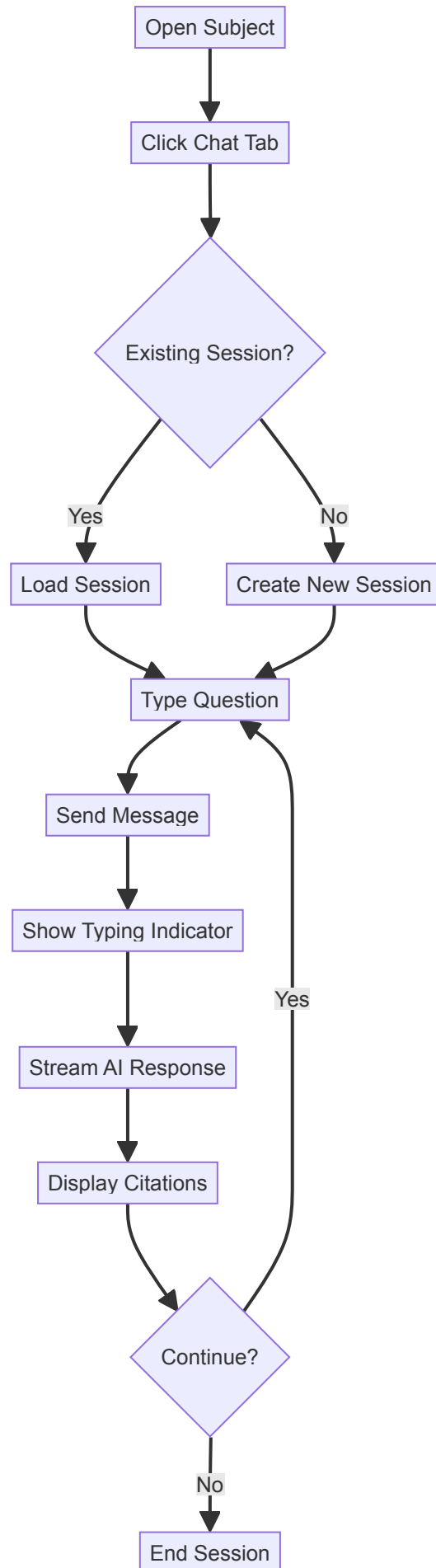


Figure 8.3: AI Chat User Flow

8.9 Responsive Design Specifications

Breakpoint	Width	Layout Changes
Desktop	1024px+	Full layout: Sidebar visible, 3-column chat, grid cards
Tablet	768px - 1023px	Collapsible sidebar (hamburger), 2-column grids, chat citations in modal
Mobile	320px - 767px	Bottom navigation bar, single column, full-screen modals, stacked stats cards

8.10 Accessibility Features

Feature	Implementation
Keyboard Navigation	All interactive elements focusable with Tab, Enter/Space to activate
Screen Reader Support	ARIA labels, roles, live regions for dynamic content (chat messages)
Color Contrast	WCAG AA compliant (4.5:1 minimum for text)
Focus Indicators	Visible focus ring (2px blue outline) on all interactive elements
Error Handling	aria-invalid, aria-describedby for form errors, focus moved to first error
Skip Links	"Skip to main content" link for keyboard users

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 Content 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

9.7 Detailed Test Cases

The following tables document specific test cases with preconditions, steps, and expected results:

9.7.1 Authentication Test Cases

TC ID	Test Case	Preconditions	Test Steps	Expected Result	Status
TC-001	Valid User Login	User account exists with email "test@example.com"	1. Navigate to /login 2. Enter email "test@example.com" 3. Enter correct password 4. Click "Sign In"	User redirected to /dashboard; JWT token stored in cookie; Welcome toast displayed	Pass
TC-002	Invalid Password Login	User account exists	1. Navigate to /login 2. Enter valid email 3. Enter wrong password 4. Click "Sign In"	Error message "Invalid email or password" displayed; User remains on login page; No token stored	Pass
TC-003	User Registration	Email "new@example.com" not registered	1. Navigate to /register 2. Fill name, email, password (strong) 3. Check terms checkbox 4. Click "Register"	Success modal shown; Verification email sent; User record created in database	Pass
TC-004	Duplicate Email Registration	Email already registered	1. Navigate to /register 2. Enter existing email 3. Complete form 4. Click "Register"	Error "Email already registered" displayed; Form not submitted	Pass
TC-005	Weak Password Rejection	On registration page	1. Enter password "123" 2. Observe password strength meter	Strength meter shows "Weak"; Submit button disabled; Requirements displayed	Pass

9.7.2 Document Management Test Cases

TC ID	Test Case	Preconditions	Test Steps	Expected Result	Status
TC-006	PDF Upload Success	User logged in; Subject selected; Valid PDF file (< 10MB)	1. Click Documents tab 2. Drag PDF to dropzone 3. Wait for upload	Progress bar shows upload; Status changes: Uploading → Processing → Complete; Document appears in list	Pass
TC-007	Invalid File Type Rejection	User logged in; Subject selected	1. Click Documents tab 2. Try to upload .exe file	Error toast "Only PDF files are allowed"; File not uploaded	Pass
TC-008	File Size Limit	User logged in; PDF file > 10MB	1. Attempt to upload large PDF	Error "File size exceeds 10MB limit"; Upload cancelled	Pass
TC-009	Syllabus Extraction	Syllabus PDF uploaded	1. Upload PDF with type "Syllabus" 2. Wait for processing 3. Click Syllabus tab	Units and topics extracted; Accordion tree displayed; Extraction accuracy >= 85%	Pass
TC-010	Document Deletion	Document exists in list	1. Click delete icon on document 2. Confirm in modal	Document removed from list; File deleted from storage; KB index updated	Pass

9.7.3 AI Chat Test Cases

TC ID	Test Case	Preconditions	Test Steps	Expected Result	Status
TC-011	Create Chat Session	User logged in; Subject with KB exists	1. Navigate to subject 2. Click Chat tab 3. Click "New Chat"	New session created; Session appears in sidebar; Input field focused	Pass
TC-012	Send Message & Get Response	Active chat session	1. Type "What is polymorphism?" 2. Press Enter or click Send	User message displayed; Typing indicator shown; AI response streams in; Citations panel updates	Pass
TC-013	Citation Links	AI response with citations	1. Click citation link in citations panel	Source document opens in new tab; Correct page displayed	Pass
TC-014	Empty Message Prevention	Active chat session	1. Leave input empty 2. Click Send button	Send button disabled; No message sent; Input field focused	Pass
TC-015	Session History Persistence	Chat session with messages	1. Close browser 2. Login again 3. Open same session	All previous messages loaded; Citations preserved; Conversation continues	Pass

9.7.4 Admin Panel Test Cases

TC ID	Test Case	Preconditions	Test Steps	Expected Result	Status
TC-016	Admin Access Control	Logged in as student	1. Try to navigate to /admin/content	Redirected to /dashboard; "Access denied" toast shown	Pass
TC-017	Create University	Logged in as admin	1. Go to /admin/content 2. Click "Add University" 3. Fill name, code, location 4. Save	University created; Appears in list; Audit log created	Pass
TC-018	View Audit Logs	Admin logged in; Actions performed	1. Navigate to /admin/audit-logs 2. Filter by action type	Logs displayed; Filter works; JSON diff visible for changes	Pass
TC-019	Delete Course with Warning	Course has associated subjects	1. Go to Courses tab 2. Click delete on course 3. View warning modal	Warning shows "This will delete X subjects"; Requires confirmation	Pass
TC-020	System Settings Update	Admin logged in	1. Go to /admin/settings 2. Change "Max File Size" to 15MB 3. Save	Setting updated; Success toast; New limit enforced on uploads	Pass

9.8 Test Environment

Component	Specification
Test Server	Docker containers on Ubuntu 22.04 LTS, 4 vCPU, 8GB RAM
Database	PostgreSQL 15.x (isolated test database, seeded data)
Cache	Redis 7.x (separate test instance)
Browser Testing	Playwright with Chrome 120+, Firefox 120+, WebKit
CI Platform	GitHub Actions with self-hosted runners
Test Data	Seeded via database migrations: 5 universities, 10 courses, 50 subjects, 100 documents

9.9 Bug Tracking

Bugs are tracked using GitHub Issues with the following classification:

Severity	Description	Response Time	Example
Critical	System down, data loss, security breach	< 4 hours	Authentication bypass, database corruption
High	Major feature broken, no workaround	< 24 hours	Chat not working, file upload failing
Medium	Feature partially broken, workaround exists	< 1 week	Export not formatting correctly
Low	Minor issue, cosmetic, enhancement	Next sprint	Typo in UI, alignment issue

9.9.1 Sample Bug Report Format

Bug ID	BUG-042
Title	Chat citations panel doesn't update on new message
Severity	Medium
Environment	Production, Chrome 120, macOS Sonoma
Steps to Reproduce	1. Open chat session 2. Send message 3. Observe citations panel
Expected	Citations panel updates with new citations
Actual	Panel shows old citations until page refresh
Root Cause	React state not updating on SSE complete event
Fix	Added useEffect dependency on citations array
Status	Resolved in v1.2.3

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/>