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.

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

ID	Category	Requirement
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

Component	App Server	Database	Cache
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

Technology	Version
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

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>Technology Used - Study in Woods</title>
   <style>
       * {
           margin: 0;
           padding: 0;
           box-sizing: border-box;
       }

       @page {
           size: A4;
           margin: 1in;
       }

       body {
           font-family: 'Times New Roman', Times, serif;
           font-size: 12pt;
           line-height: 1.6;
           color: #000;
       }

       h1 {
           text-align: center;
           font-size: 16pt;
           font-weight: bold;
           text-decoration: underline;
           margin: 20px 0 30px 0;
           text-transform: uppercase;
       }

       h2 {
           font-size: 14pt;
           font-weight: bold;
           margin-top: 25px;
           margin-bottom: 15px;
           text-decoration: underline;
       }

       h3 {
           font-size: 12pt;
           font-weight: bold;
           margin-top: 20px;
```

```
    margin-bottom: 10px;
    font-style: italic;
}

p {
    text-align: justify;
    margin-bottom: 12px;
    text-indent: 0.5in;
}

p.no-indent {
    text-indent: 0;
}

ul, ol {
    margin-left: 0.75in;
    margin-bottom: 12px;
}

li {
    margin-bottom: 8px;
    text-align: justify;
}

.page-break {
    page-break-after: always;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin: 20px 0;
}

th, td {
    border: 1px solid #000;
    padding: 8px;
    text-align: left;
    vertical-align: top;
}

th {
    background-color: #f0f0f0;
    font-weight: bold;
}

.page-number {
```

```
           text-align: center;
           font-size: 11pt;
           margin-top: 30px;
       }

       .architecture-box {
           border: 2px solid #000;
           padding: 15px;
           margin: 20px 0;
           font-family: 'Courier New', monospace;
           font-size: 10pt;
           background-color: #f9f9f9;
       }
   </style>
</head>
<body>
   <h1>4. TECHNOLOGY USED</h1>

   <h2>4.1 Technology Stack Overview</h2>

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

   <h2>4.2 Frontend Technologies</h2>

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

   <div class="page-break"></div>

   <h2>4.3 Backend Technologies</h2>

   <table>
       <thead>
           <tr>
               <th>Library</th>
               <th>Version</th>
               <th>Purpose</th>
               <th>Key Capabilities</th>
           </tr>
       </thead>
       <tbody>
           <tr>
               <td>Go</td>
               <td>1.24.1</td>
               <td>Programming Language</td>
               <td>Goroutines, Channels, Fast Compilation,
GC</td>
           </tr>
           <tr>
               <td>Fiber</td>
               <td>2.52.5</td>
               <td>Web Framework</td>
               <td>Fasthttp, Middleware, Routing, Context</td>
           </tr>
           <tr>
               <td>GORM</td>
               <td>1.31.0</td>
               <td>ORM</td>
               <td>Migrations, Associations, Hooks,
Preloading</td>
           </tr>
           <tr>
```

```
               <td>JWT</td>
               <td>5.3.0</td>
               <td>Authentication</td>
               <td>Token Generation, RS256, Claims
Validation</td>
           </tr>
           <tr>
               <td>bcrypt</td>
               <td>0.43.0</td>
               <td>Password Hashing</td>
               <td>Adaptive Cost, Automatic Salting</td>
           </tr>
           <tr>
               <td>go-redis</td>
               <td>9.16.0</td>
               <td>Redis Client</td>
               <td>Connection Pooling, Pipelining, Pub/Sub</td>
           </tr>
           <tr>
               <td>AWS SDK</td>
               <td>1.55.8</td>
               <td>S3 Client</td>
               <td>Multipart Upload, Pre-signed URLs,
Retries</td>
           </tr>
           <tr>
               <td>Validator</td>
               <td>10.28.0</td>
               <td>Input Validation</td>
               <td>Struct Tags, Custom Validators, Error
Messages</td>
           </tr>
           <tr>
               <td>Cron</td>
               <td>3.0.1</td>
               <td>Job Scheduling</td>
               <td>Cron Expressions, Job Chains, Error Handling</
td>
           </tr>
       </tbody>
   </table>

   <h2>4.4 Database Technologies</h2>

   <table>
       <thead>
           <tr>
```

```
               <th>Technology</th>
               <th>Version</th>
               <th>Type</th>
               <th>Primary Use Cases</th>
           </tr>
       </thead>
       <tbody>
           <tr>
               <td>PostgreSQL</td>
               <td>15.x</td>
               <td>Relational Database</td>
               <td>Permanent data storage, Complex queries, ACID
transactions, JSONB support</td>
           </tr>
           <tr>
               <td>Redis</td>
               <td>7.x</td>
               <td>In-Memory Cache</td>
               <td>Session storage, Rate limiting, Temporary
data, Pub/Sub</td>
           </tr>
       </tbody>
   </table>

   <div class="page-break"></div>

   <h2>4.5 Cloud Services & Infrastructure</h2>

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

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

   <h2>4.6 Development & Deployment Tools</h2>

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

   <div class="page-break"></div>

   <h2>4.7 System Architecture</h2>

   <div class="architecture-box">
                        CLIENT LAYER
(Next.js 15 + React 19 + TypeScript)


                   HTTPS/TLS Connection

                   LOAD BALANCER (DO)
```

```
SSL Termination, Health Checks

                          API Instance 1    API Instance 2    API
Instance N     (Go + Fiber)      (Go + Fiber)      (Go + Fiber)



  PostgreSQL       Redis      DO Spaces   DO AI       Database
Cache        Storage   GradientAI
Sessions     LLM       Rate Limit  KB API  Progress     RAG
Cache
Chat History
   </div>

   <h2>4.8 Technology Selection Summary</h2>

   <table>
       <thead>
           <tr>
               <th>Decision</th>
               <th>Choice</th>
               <th>Rationale</th>
           </tr>
       </thead>
       <tbody>
           <tr>
               <td>Backend Language</td>
               <td>Go over Node.js/Python</td>
               <td>3x more requests/sec than Node.js, single 
binary deployment, compile-time error checking</td>
           </tr>
           <tr>
               <td>Database</td>
               <td>PostgreSQL over MongoDB</td>
               <td>Relational data model fits academic hierarchy, 
ACID guarantees, JSONB for flexibility</td>
           </tr>
           <tr>
               <td>Frontend Framework</td>
               <td>Next.js over CRA</td>
               <td>SSR for SEO, 60% faster initial load, built-in 
routing and image optimization</td>
           </tr>
           <tr>
               <td>Cloud Provider</td>
               <td>DigitalOcean over AWS</td>
               <td>Transparent pricing, Bangalore data center 
```

```
(15-30ms latency), integrated GradientAI</td>
           </tr>
       </tbody>
   </table>
   
   <div class="page-number">4</div>
</body>
</html>
```

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:	3-4	JWT auth, login/register,	

Phase	Weeks	Key Deliverables	LOC
Authentication		password reset	
3: Academic Hierarchy	5-6	University/Course/Semester/ Subject CRUD	
4: Document Upload	7-8	DigitalOcean Spaces integration, multipart upload	
5: AI Knowledge Base	9-10	GradientAI KB integration, document indexing	~4,200
6: Syllabus Extraction	11-12	Llama 3.3 70B extraction, structured JSON output	
7: Chat Sessions	13-14	Session CRUD, message history	
8: AI Chat	15-16	SSE streaming, KB-context responses	~3,800
9: Citations	17-18	Source document references in responses	
10: PYQ Extraction	19-20	Question extraction from exam papers	
11: Analytics	21-22	Usage tracking, dashboard charts	~3,000
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 + React 19 + TypeScript) │ │ ┌──────────────┐ ┌──────────────┐
```

```
┌──────────────┐ ┌────────────┐ │ │ │ Pages │ │ Components │ │ Hooks │ │
```

```
State │ │ │ │ (Routing) │ │ (shadcn) │ │ (React Hook) │ │ (TanStack)│ │ │
```

```
└──────────────┘   └──────────────┘   └──────────────┘   └────────────┘   │
```

```
└───────────────────────────────┬──────────────────────────────────────┘ │
```

```
HTTPS/REST          API          +          SSE          ▼

┌──────────────────────────────────────────────────────────┐ │

APPLICATION  LAYER │ │ (Go  1.24  +  Fiber  2.52) │ │ ┌────────────┐

┌────────────┐ ┌────────────┐ ┌──────────┐ │ │ │ Handlers │ │ Services

│ │ Middleware │ │ Utils │ │ │ │ (API Routes)│ │ (Business) │ │ (Auth,CORS)

│ │ (Crypto) │ │ │ └────────────┘ └────────────┘ └────────────┘

└──────────┘                                                   │

└─────────────┬───────────┬──────────┬───────────┬─────────┘ ▼

▼   ▼   ▼   ┌────────────────┐   ┌───────────┐   ┌────────────┐

┌────────────┐ │ PostgreSQL  15 │ │ Redis  7 │ │ DO  Spaces │ │ DO

GradientAI│ │ (30+ Tables) │ │ (Cache/TTL) │ │ (S3 Storage) │ │ (Llama 3.3)
```

```
│ └───────────────────┘ └─────────────┘ └──────────────┘ └──────────────┘
```

6.2 Design Patterns Used

- **Repository Pattern:** GORM-based data access layer abstracts database operations
- **Service Layer Pattern:** 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)

```
┌────────────────┐ Login, Upload, Questions ┌──────────────┐ │ Student
```

```
│───────────────────────────────>│     Study     in     │     │     User
```

```
│<───────────────────────────────│ Woods │ └──────────┘ Auth Tokens, AI
```

```
Responses └──────┬───────┘ │ ┌──────────┐ Manage Users, Config │ │ Admin
```

```
├──────────────────────────────────────────┘ └──────────┘ Audit Logs, Reports
```

```
│ ┌──────────────────┼──────────────────┐ ▼ ▼ ▼ ┌──────────────┐ ┌──────────┐
```

```
┌──────────────────┐ │ DO AI API │ │ Database │ │ Email Service │
```

```
└──────────────┘ └──────────┘ └──────────────────┘
```

6.3.2 System Decomposition (Level 1)

```
Student ──> [1.0 Authentication] ──> User Database │ ├──> [2.0 Academic
```

```
Management] ──> University/Course/Subject DB ├──> [3.0 Document Processing]
```

```
──> DO Spaces + AI Extraction ├──> [4.0 AI Chat System] ──> Knowledge Base +
```

```
AI Service └──> [5.0 Analytics] ──> Activity Logs Admin ──> [6.0
```

```
Administration] ──> All Databases + Audit Logs
```

6.4 Entity Relationship Diagram

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

```
┌───────────────┐  ┌───────────────┐  │ User  │  │ University
```

```
│ │ PK id │ │ PK id │ │ email │ │ name │ │ role │ │ code │ └────┬──────┘
```

```
└────┬──────┘  │ 1:N │ 1:N │ ▼ │ ┌──────────┐  ┌──────────┐  │ │
```

```
Course  │──1:N──>│  Semester  │  │  │  FK  univ_id  │  │  FK  course_id  │  │
```

```
└──────────┘  └────┬──────┘  │ │ 1:N │ ▼ │ ┌──────────┐  │ │ Subject
```

```
│ │ │ FK  semester │ │ │ kb_uuid │ │ └──┬───────┘ │ 1:N │ 1:N ▼ ▼
```

```
┌──────────┐  ┌──────────┐  │  ChatSession  │  │  Document  │──1:N──>
```

```
Syllabus  │  FK  user_id  │  │  FK  subject  │  │  │  FK  subject  │  │  status  │
```

```
SyllabusUnit  └────┬──────┘  └──────────┘  │  │  1:N  SyllabusTopic  ▼
```

```
┌──────────────┐ Other: APIKey, UserActivity, AdminAuditLog, │ ChatMessage │
```

```
AppSetting, PYQ, ChatMemory, IndexingJob │ FK session │ │ citations │
```

```
└──────────────┘
```

6.5 Key Sequence Flows

6.5.1 Authentication Flow

```
User -> Frontend -> POST /login -> Validate -> DB Lookup -> bcrypt Check │
```

```
<── JWT Token ── Generate JWT <─┘ │ Store Session in Redis (24h TTL)
```

6.5.2 Document Upload & AI Extraction

```
User -> Upload PDF -> Validate -> Upload to DO Spaces -> Save Metadata
```

```
(pending) │ [Background] ▼ AI Extract Syllabus -> Parse JSON -> Store Units/
```

```
Topics │ Index to Knowledge Base -> Poll Status -> Update (completed)
```

6.5.3 AI Chat with RAG

```
User  Question  —>  Get  Session  History  —>  Query  Knowledge  Base  —>  Get
```

```
Citations │ <── SSE Stream ── AI Response <──┘ │ Save Messages + Citations
```

```
to DB
```

6.6 Component Architecture

6.6.1 Backend Layers

```
┌──────────────────────────────────────────────────────────────────────
```

```
───────────────────────────┐ │ API  Layer:  Auth  |  Course  |  Document  |
```

```
Chat                 |              Admin              Handlers            │
```

```
├─────────────────────────────────────────────────────────────┤         │
```

```
Middleware:  JWT  Verify  |  CORS  |  Rate  Limit  (Redis)  |  Logger  │
```

```
├──────────────────────────────────────────────────────────┤ │
```

```
Service Layer: AuthService | SyllabusExtractor | ChatService │
```

```
├──────────────────────────────────────────────────────────┤ │
```

```
Repository Layer: User | Subject | Document | Chat (GORM) │
```

```
├──────────────────────────────────────────────────────────┤ │ Data:
```

```
PostgreSQL (30+ tables) | Redis (Cache/Sessions) │
```

```
├──────────────────────────────────────────────────────────┤ │
```

```
External: DO Spaces (S3) | DO GradientAI (LLM/KB) | SMTP │
```

```
└──────────────────────────────────────────────────────────┘
```

6.6.2 Frontend Structure

```
App (Next.js Router) ├── Layout: Header | Sidebar | Footer ├── Pages │ ├──
```

```
Auth: /login, /register │ ├── Dashboard: Stats, Activity, Actions │ ├──
```

```
Academic: /universities, /courses, /subjects │ ├── Chat: /chat/[subjectId] -
```

```
Sessions, Messages, Citations │ ├── Analytics: Charts, Stats, Activity Table
```

```
│ └── Admin: Users, Settings, Audit Logs ├── Providers: QueryProvider |
```

```
ThemeProvider | AuthProvider └── Utils: API Client (Axios) | Hooks |
```

```
Validators (Zod)
```

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 &	api_keys, api_key_usage_logs,	API management, activity

Category	Tables	Purpose
Audit	user_activities, admin_audit_logs, app_settings	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

Column	Type	Key/Constraint
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',

Column	Type	Purpose
		'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.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Testing - Study in Woods</title>
   <style>
       * { margin: 0; padding: 0; box-sizing: border-box; }
       @page { size: A4; margin: 1in; }
       body { font-family: 'Times New Roman', Times, serif; font-
size: 12pt; line-height: 1.6; color: #000; }
       h1 { text-align: center; font-size: 16pt; font-weight:
bold; text-decoration: underline; margin: 20px 0 30px 0; text-
transform: uppercase; }
       h2 { font-size: 14pt; font-weight: bold; margin-top: 25px;
margin-bottom: 15px; text-decoration: underline; }
       h3 { font-size: 12pt; font-weight: bold; margin-top: 20px;
margin-bottom: 10px; font-style: italic; }
       p { text-align: justify; margin-bottom: 12px; text-indent:
0.5in; }
       p.no-indent { text-indent: 0; }
       ul, ol { margin-left: 0.75in; margin-bottom: 12px; }
       li { margin-bottom: 8px; text-align: justify; }
       .page-break { page-break-after: always; }
       table { width: 100%; border-collapse: collapse; margin:
20px 0; }
       th, td { border: 1px solid #000; padding: 8px; text-align:
left; vertical-align: top; }
       th { background-color: #f0f0f0; font-weight: bold; }
   </style>
</head>
<body>
   <h1>9. TESTING</h1>

   <h2>9.1 Testing Strategy</h2>

   <p>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%).</p>

   <h2>9.2 Types of Testing</h2>

   <ul>
       <li><strong>Unit Testing:</strong> Go (testify) - 156
tests covering services, handlers, utilities; React (Jest/RTL) -
89 component and hook tests</li>
       <li><strong>Integration Testing:</strong> API tests with
```

```
real PostgreSQL/Redis via Docker; database relationship and
constraint verification</li>
       <li><strong>End-to-End Testing:</strong> Playwright
browser automation for critical user journeys</li>
       <li><strong>Security Testing:</strong>
Authentication/authorization verification, input validation (SQL
injection, XSS, path traversal), dependency scanning</li>
       <li><strong>Performance Testing:</strong> Load testing
with k6 (100-1000 concurrent users), API response time
benchmarks, database query optimization</li>
   </ul>

   <h2>9.3 Test Cases Summary</h2>

   <table>
       <thead>
           <tr>
               <th>Test Suite</th>
               <th>Cases</th>
               <th>Coverage Area</th>
           </tr>
       </thead>
       <tbody>
           <tr><td>Authentication</td><td>12</td><td>Register,
Login, JWT, Password reset</td></tr>
           <tr><td>Academic
Hierarchy</td><td>23</td><td>Universities, Courses, Subjects,
Relationships</td></tr>
           <tr><td>Documents &
Syllabus</td><td>28</td><td>Upload, Validation, Extraction,
Storage</td></tr>
           <tr><td>Chat & AI</td><td>14</td><td>Sessions,
Messages, Streaming, Citations</td></tr>
           <tr><td>Admin</td><td>9</td><td>User management,
Settings, Audit logs</td></tr>
       </tbody>
   </table>

   <h2>9.4 E2E User Journeys</h2>

   <table>
       <thead>
           <tr>
               <th>User Journey</th>
               <th>Steps</th>
               <th>Duration</th>
               <th>Pass Rate</th>
           </tr>
       </thead>
       <tbody>
           <tr><td>User Registration &
Login</td><td>7</td><td>15s</td><td>98%</td></tr>
           <tr><td>Course
Enrollment</td><td>10</td><td>22s</td><td>96%</td></tr>
           <tr><td>Document Upload &
Processing</td><td>12</td><td>45s</td><td>94%</td></tr>
           <tr><td>AI Chat
Interaction</td><td>8</td><td>18s</td><td>97%</td></tr>
           <tr><td>Admin User
Management</td><td>9</td><td>20s</td><td>99%</td></tr>
       </tbody>
   </table>

   <h2>9.5 Performance Benchmarks</h2>

   <table>
       <thead>
           <tr>
               <th>Test Type</th>
               <th>Virtual Users</th>
               <th>Duration</th>
               <th>Result</th>
           </tr>
       </thead>
       <tbody>
           <tr><td>Baseline Load</td><td>100</td><td>5
min</td><td>95% requests < 2s</td></tr>
           <tr><td>Spike Test</td><td>50 Pass</td></tr>
           <tr><td>Integration Test
Coverage</td><td>60%</td><td>68%</td><td> Pass</td></tr>
           <tr><td>API Response Time (p95)</td><td><
2s</td><td>1.2s</td><td> Pass</td></tr>
           <tr><td>CI Pipeline Duration</td><td>< 15
min</td><td>10 min</td><td> Pass</td></tr>
       </tbody>
   </table>

   <div class="page-number">9</div>
</body>
</html>
```

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/
10