3. REQUIREMENTS

3.1 Functional Requirements

Functional requirements define the specific behaviors, functions, and capabilities that the Study in Woods system must provide to its users. These requirements encompass all user-facing features, data processing capabilities, and system operations necessary for the platform to fulfill its educational objectives.

3.1.1 User Management and Authentication

FR-1: User Registration

The system shall allow new users to register by providing their full name, email address, and password. The registration process must validate email format, enforce password strength requirements (minimum 8 characters with uppercase, lowercase, numbers, and special characters), and prevent duplicate email registrations.

FR-2: User Authentication

The system shall authenticate users using JWT (JSON Web Token) based authentication. Upon successful login, the system must generate a secure access token with configurable expiration time and maintain session state using Redis cache for improved performance and security.

FR-3: Password Management

The system shall provide password reset functionality through email verification. Users must be able to request password reset links, which shall expire after 24 hours. All passwords must be hashed using bcrypt algorithm with a cost factor of 12 or higher before storage.

FR-4: Role-Based Access Control

The system shall implement role-based access control with two primary roles: Student and Admin. Admin users must have elevated privileges including user management, system configuration, and access to audit logs, while Student users have access limited to their own academic content.

3.1.2 Academic Hierarchy Management

FR-5: University Management

The system shall enable administrators to create, read, update, and delete university records. Each university must store metadata including name, location, website URL, and status (active/inactive). The system must support multiple universities within a single instance.

FR-6: Course Management

The system shall allow creation and management of academic courses associated with universities. Each course must specify program name, duration, total semesters, and course code. The system must maintain parent-child relationships between universities and courses.

FR-7: Semester Management

The system shall automatically generate semester records based on course duration. Each semester must be numbered sequentially and associated with a specific course. The system must support dynamic semester creation when course details are modified.

FR-8: Subject Management

The system shall enable creation and management of subjects within semesters. Each subject must store subject code, name, credits, description, and syllabus content. The system must maintain hierarchical relationships: University → Course → Semester → Subject.

3.1.3 Document Management and Processing

FR-9: Document Upload

The system shall allow users to upload study materials in PDF format with maximum file size of 10MB. The upload process must validate file type, size, and integrity before accepting documents. Files shall be stored in DigitalOcean Spaces cloud storage with unique identifiers.

FR-10: Syllabus Extraction

The system shall automatically extract structured information from uploaded syllabus PDFs using AI-powered natural language processing. The extraction must identify course units, topics, subtopics, teaching hours, and learning outcomes with a minimum accuracy of 85%.

FR-11: Document Indexing

The system shall automatically index uploaded documents into subject-specific AI knowledge bases. The indexing process must parse document content, create embeddings, and make content searchable through the AI chat interface within 2 minutes of upload completion.

FR-12: Document Retrieval

The system shall provide document download functionality with pre-signed URLs that expire after 1 hour. Users must be able to view document metadata, download status, and access their uploaded files securely without exposing permanent storage URLs.

3.1.4 AI-Powered Features

FR-13: AI Chat Interface

The system shall provide an intelligent conversational interface powered by Llama 3.3 70B language model through DigitalOcean's GradientAI platform. The chat interface must support multi-turn conversations, maintain context across messages, and provide responses within 5 seconds for 90% of queries.

FR-14: Knowledge Base Integration

The system shall create and maintain subject-specific AI knowledge bases that store indexed documents. Each subject must have a dedicated knowledge base that enables contextual responses based on uploaded course materials. The system must support up to 100 documents per knowledge base.

FR-15: Context-Aware Responses

The AI chat system shall generate responses using both the language model's general knowledge and specific content from indexed documents. Responses must cite relevant documents when answering questions about course materials and clearly indicate when information is not available in the knowledge base.

FR-16: Session Management

The system shall maintain chat sessions with persistent conversation history. Each session must store all messages, timestamps, token usage, and response times. Users must be able to create new sessions, view previous sessions, and delete unwanted conversations.

3.1.5 Past Year Questions (PYQ) Management

FR-17: PYQ Extraction

The system shall extract questions from uploaded examination paper PDFs using AI-powered text analysis. The extraction process must identify individual questions, categorize them by subject topics, and assign difficulty levels (Easy, Medium, Hard) based on complexity analysis.

FR-18: Question Categorization

The system shall automatically categorize extracted questions by linking them to relevant syllabus units and topics. The categorization must achieve minimum 80% accuracy and allow manual correction by users. Questions must be tagged with exam year, semester, and question type.

3.1.6 Analytics and Monitoring

FR-19: Activity Tracking

The system shall track all user activities including document uploads, chat interactions, subject accesses, and system feature usage. Activity logs must record timestamps, user identifiers, resource IDs, and action types for comprehensive analytics and audit purposes.

FR-20: Usage Statistics

The system shall generate real-time usage statistics including total users, active sessions, document count, chat message volume, and API call frequency. Statistics must be aggregated hourly and stored for historical trend analysis spanning at least 90 days.

3.1.7 API Access and Integration

FR-21: API Key Management

The system shall enable users to generate encrypted API keys for programmatic access to system resources. API keys must support scope-based permissions (read, write, admin), IP whitelisting, rate limiting, and automatic expiration. All API keys must be encrypted using AES-256 encryption before database storage.

FR-22: RESTful API

The system shall provide a comprehensive RESTful API with 100+ documented endpoints covering all major functionalities. The API must follow REST conventions, return appropriate HTTP status codes, and provide detailed error messages with error codes for debugging.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

NFR-1: Response Time

The system shall respond to 95% of user requests within 2 seconds under normal load conditions. Database queries must execute within 500 milliseconds, and API endpoints must return responses within 1 second for read operations. AI chat responses must stream within 5 seconds of request initiation.

NFR-2: Throughput

The system shall support minimum 1000 concurrent users without performance degradation. The backend API must handle at least 10,000 requests per minute across all endpoints. Document upload processing must support parallel uploads of up to 50 files simultaneously.

NFR-3: Resource Utilization

The application shall maintain CPU usage below 70% under normal load conditions and memory consumption below 2GB per instance. Database connection pooling must limit concurrent connections to 100 to prevent resource exhaustion. Redis cache hit ratio must exceed 80% for frequently accessed data.

3.2.2 Security Requirements

NFR-4: Authentication Security

The system shall implement secure authentication using industry-standard JWT tokens with RS256 signing algorithm. Tokens must expire after configurable periods (default: 24 hours for access tokens, 7 days for refresh tokens). The system must enforce HTTPS for all client-server communications.

NFR-5: Data Encryption

All sensitive data including passwords and API keys must be encrypted at rest using AES-256 encryption. Password hashing must use bcrypt with minimum cost factor of 12. Database connections must use SSL/TLS encryption for data in transit.

NFR-6: Access Control

The system shall implement role-based access control (RBAC) with principle of least privilege. Administrative functions must require elevated permissions verified on every request. The system must prevent privilege escalation attacks through proper authorization checks.

NFR-7: Protection Against Attacks

The system shall implement rate limiting to prevent brute force attacks (maximum 5 failed login attempts per 15 minutes). CORS policies must restrict API access to whitelisted domains. Input validation must prevent SQL injection, XSS, and CSRF attacks. The system must log all security-related events for audit purposes.

3.2.3 Usability Requirements

NFR-8: User Interface

The web interface shall be intuitive and require minimal training for users with basic computer literacy. Navigation must follow consistent patterns across all pages. The system must provide contextual help and error messages that clearly explain issues and suggest corrective actions.

NFR-9: Responsive Design

The user interface shall be fully responsive and functional across desktop (1920x1080), tablet (1024x768), and mobile (375x667) screen sizes. The system must maintain readability and usability on all supported devices without horizontal scrolling or layout breaks.

NFR-10: Accessibility

The system shall comply with WCAG 2.1 Level AA accessibility guidelines. The interface must support keyboard navigation, screen readers, and provide sufficient color contrast (minimum 4.5:1 ratio). Form inputs must have proper labels and error states must be clearly indicated.

3.2.4 Reliability Requirements

NFR-11: Availability

The system shall maintain 99.5% uptime excluding planned maintenance windows. Planned maintenance must be scheduled during low-usage periods (2:00 AM - 5:00 AM IST) with minimum 48-hour advance notice to users. The system must implement health check endpoints that respond within 100 milliseconds.

NFR-12: Fault Tolerance

The system shall gracefully handle failures in external services (AI API, cloud storage) without complete system failure. When DigitalOcean services are unavailable, the system must queue requests and retry with exponential backoff. Database connection failures must trigger automatic reconnection attempts.

NFR-13: Data Integrity

The system shall maintain ACID compliance for all database transactions. Data consistency must be verified through foreign key constraints and validation rules. Automated daily backups must be performed with point-in-time recovery capability up to 30 days.

3.2.5 Scalability Requirements

NFR-14: Horizontal Scaling

The application architecture shall support horizontal scaling by adding multiple backend instances behind a load balancer. The system must maintain stateless API design to enable seamless distribution of requests across multiple servers. Session state must be externalized to Redis for shared access.

NFR-15: Database Scalability

The PostgreSQL database shall support read replicas for query distribution. The system must implement connection pooling with configurable pool sizes (default: 25-100 connections).

Database indexes must be optimized for common query patterns to maintain performance as data volume grows.

NFR-16: Storage Scalability

The DigitalOcean Spaces storage shall scale automatically to accommodate growing document volumes. The system must support storage of up to 10,000 documents initially with capability to expand to 100,000 documents without architectural changes. CDN integration must ensure fast document delivery globally.

3.2.6 Maintainability Requirements

NFR-17: Code Quality

The codebase shall maintain clean architecture with clear separation of concerns across layers (handlers, services, repositories). Code must follow language-specific style guides (Go: gofmt, TypeScript: ESLint) and maintain minimum 70% test coverage for critical business logic.

NFR-18: Documentation

All API endpoints must be documented with request/response examples, parameter descriptions, and error codes. Code must include inline comments for complex logic and business rules. System architecture and deployment procedures must be documented in markdown format.

NFR-19: Logging and Monitoring

The system shall implement comprehensive logging with configurable log levels (DEBUG, INFO, WARN, ERROR). All errors must be logged with stack traces and contextual information. The system must expose Prometheus-compatible metrics for monitoring CPU, memory, request rates, and response times.

3.3 Hardware Requirements

3.3.1 Client-Side Requirements

Users accessing the Study in Woods platform must have computing devices that meet the following minimum specifications to ensure optimal performance and user experience.

Component	Minimum Requirement	Recommended Requirement
Processor	Dual-core 1.6 GHz or equivalent	Quad-core 2.4 GHz or higher
RAM	2 GB	4 GB or higher
Storage	500 MB available space	1 GB available space
Display	1024 x 768 resolution	1920 x 1080 resolution
Network	1 Mbps internet connection	5 Mbps or higher broadband
Browser	Chrome 90+, Firefox 88+, Safari 14+, Edge 90+	Latest version of supported browsers

3.3.2 Server-Side Requirements

The Study in Woods backend infrastructure requires the following hardware specifications for production deployment. These specifications support 1000+ concurrent users with optimal performance.

Component	Production Server	Database Server	Redis Cache Server
Processor	4 vCPU (Intel Xeon or AMD EPYC)	4 vCPU (Intel Xeon or AMD EPYC)	2 vCPU
RAM	8 GB DDR4	8 GB DDR4	4 GB DDR4
Storage	80 GB SSD (NVMe preferred)	200 GB SSD (NVMe preferred)	40 GB SSD
Network	1 Gbps bandwidth	1 Gbps bandwidth	1 Gbps bandwidth
Operating System	Ubuntu 22.04 LTS or Docker-compatible Linux	Ubuntu 22.04 LTS	Ubuntu 22.04 LTS

Cloud Infrastructure Specifications:

- **DigitalOcean Droplet:** Premium Intel (4 vCPU, 8 GB RAM) or comparable instance type
- **DigitalOcean Spaces:** Minimum 100 GB storage capacity with CDN enabled for Bangalore (BLR1) region
- **Load Balancer:** DigitalOcean Load Balancer or equivalent with SSL termination and health checks
- **Backup Storage:** Minimum 50 GB for database backups with 30-day retention

3.4 Software Requirements

3.4.1 Frontend Software Stack

Core Framework and Libraries:

Technology	Version	Purpose
Next.js	15.5.6	React framework with server-side rendering, routing, and optimization
React	19.1.0	UI library for building interactive user interfaces
TypeScript	5.x	Type-safe JavaScript for better development experience
Tailwind CSS	4.0	Utility-first CSS framework for responsive design
shadcn/ui	Latest	Accessible UI component library built on Radix UI
Framer Motion	12.23.24	Animation library for smooth UI transitions
TanStack Query	5.90.9	Data fetching, caching, and state management
Axios	1.13.2	HTTP client for API communication
React Hook Form	7.66.0	Form validation and state management
Zod	4.1.12	Schema validation for forms and API responses

3.4.2 Backend Software Stack

Core Technologies:

Technology	Version	Purpose
Go (Golang)	1.24.1	Primary backend programming language
Fiber	2.52.5	High-performance web framework for Go
GORM	1.31.0	ORM library for database operations
PostgreSQL Driver	1.6.0	GORM PostgreSQL driver
JWT	5.3.0	JSON Web Token authentication
bcrypt	0.43.0	Password hashing and verification
go-redis	9.16.0	Redis client for caching and session management
AWS SDK	1.55.8	S3-compatible client for DigitalOcean Spaces
Validator	10.28.0	Struct and field validation
Cron	3.0.1	Background job scheduling

3.4.3 Database and Storage

Component	Version	Purpose
PostgreSQL	15.x	Primary relational database for structured data
Redis	7.x	In-memory cache for sessions, rate limiting, and temporary data
DigitalOcean	Latest	S3-compatible object storage for PDF

Component	Version	Purpose
Spaces		documents

3.4.4 Development and Deployment Tools

Tool	Version	Purpose
Docker	24.0+	Containerization platform for consistent deployments
Docker Compose	2.x	Multi-container orchestration for local development
Air	Latest	Live reload for Go development
Turbopack	Integrated	Fast bundler for Next.js development
GitHub Actions	Latest	CI/CD pipeline for automated testing and deployment
Git	2.x	Version control system

3.4.5 Cloud Services and APIs

Service	Provider	Purpose
Compute (Droplets)	DigitalOcean	Virtual private servers for application hosting
Object Storage (Spaces)	DigitalOcean	Scalable storage for PDF documents with CDN

Service	Provider	Purpose
GradientAI Platform	DigitalOcean	AI model hosting and inference (Llama 3.3 70B)
Knowledge Bases	DigitalOcean AI	Vector database for document indexing and retrieval
Load Balancer	DigitalOcean	Traffic distribution and SSL termination

3.5 External Interface Requirements

3.5.1 User Interface Requirements

UI-1: Login Interface

The login page shall present a centered form containing email and password input fields with appropriate labels. The form must include a "Remember Me" checkbox, "Forgot Password" link, and a prominent "Sign In" button. Input validation errors must be displayed inline below each field with red text. The interface must be responsive and centered on all screen sizes.

UI-2: Dashboard Interface

The main dashboard shall display a sidebar navigation menu with icons and labels for Universities, Courses, Subjects, Documents, Chat, and Analytics sections. The content area must show relevant cards and statistics based on the selected section. The interface must include a header with user profile dropdown, notifications icon, and theme toggle button.

UI-3: Document Upload Interface

The document upload interface shall provide a drag-and-drop zone with clear visual feedback for file selection. The interface must display upload progress with percentage completion and estimated time remaining. Uploaded files must appear in a list showing filename, size, upload date, and status (Processing, Indexed, Failed) with appropriate icons.

UI-4: Chat Interface

The chat interface shall display a conversation view with distinct visual styling for user messages (right-aligned, blue background) and AI responses (left-aligned, gray background). The interface must include a text input area with send button, session selector dropdown, and new session button. AI responses must support markdown rendering including code blocks, lists, and emphasis.

UI-5: Admin Interface

The admin panel shall provide tabbed sections for Users, Analytics, Settings, and Audit Logs. User management must display a searchable table with columns for Name, Email, Role, Status, and Actions (Edit, Delete, Reset Password). Each action must trigger a confirmation modal before execution. The interface must show comprehensive system statistics with interactive charts using Recharts library.

3.5.2 API Interface Requirements

API-1: RESTful Architecture

All API endpoints shall follow RESTful conventions using appropriate HTTP methods (GET for retrieval, POST for creation, PUT for updates, DELETE for removal). URLs must follow the pattern /api/v1/{resource}/{id}/{sub-resource} for hierarchical resources. All endpoints must accept and return JSON-formatted data with Content-Type: application/json headers.

API-2: Authentication Header

Protected API endpoints shall require a Bearer token in the Authorization header with format "Bearer {jwt_token}". The system must validate token signature, expiration, and user permissions before processing requests. Invalid or expired tokens must return HTTP 401 Unauthorized with error details in JSON format.

API-3: Request Validation

All API requests shall validate input parameters for type correctness, required fields, format constraints, and business rules. Validation failures must return HTTP 400 Bad Request with a structured error response containing field-specific error messages. The response format must include error code, message, and failed field names.

API-4: Response Format

Successful API responses shall return HTTP 200-299 status codes with a consistent JSON structure containing "success" boolean, "data" object, and optional "message" string. Error responses must include "success: false", "error" object with code and message, and HTTP 400-599 status codes based on error type. Paginated responses must include metadata for total count, page number, and page size.

API-5: Rate Limiting Headers

All API responses shall include rate limiting headers: X-RateLimit-Limit (total allowed requests), X-RateLimit-Remaining (requests remaining), and X-RateLimit-Reset (timestamp for limit reset). When rate limit is exceeded, the API must return HTTP 429 Too Many Requests with Retry-After header indicating wait time in seconds.

3.5.3 DigitalOcean AI Platform Interface

AI-1: Knowledge Base API

The system shall interface with DigitalOcean Knowledge Base API to create subject-specific knowledge bases, upload and index documents, and manage knowledge base lifecycle. API calls must include authentication via DigitalOcean API token in X-Auth-Token header. The system must handle asynchronous indexing operations with polling for completion status.

AI-2: Chat Completion API

The system shall interface with DigitalOcean GradientAI Chat Completion API using the OpenAI-compatible format. Requests must specify model name (llama-3.3-70b-instruct), messages array with role and content, and knowledge base ID for context. The system must handle streaming responses using Server-Sent Events (SSE) for real-time message delivery.

AI-3: Model Configuration

AI API requests shall include configuration parameters: temperature (0.7 default for balanced creativity), max_tokens (4096 default), top_p (0.9 for nucleus sampling), and frequency_penalty (0.0 default). The system must respect token limits and split large requests when necessary. Response metadata must be captured including token usage, model version, and processing time.

3.5.4 Storage Interface Requirements

STORAGE-1: DigitalOcean Spaces Interface

The system shall interface with DigitalOcean Spaces using S3-compatible API through AWS SDK for Go. Upload operations must use multipart upload for files larger than 5MB with 5MB chunk size. All objects must be stored with private ACL and accessed through pre-signed URLs with 1-hour expiration. The system must set appropriate Content-Type headers based on file extensions.

STORAGE-2: CDN Integration

Uploaded documents shall be accessible through DigitalOcean Spaces CDN for improved delivery performance. The CDN must cache objects based on Cache-Control headers with maximum age of 3600 seconds for static documents. The system must support cache invalidation through API calls when documents are updated or deleted.

3.5.5 Communication Protocols

PROTOCOL-1: HTTPS/TLS

All client-server communication shall use HTTPS protocol with TLS 1.2 or higher encryption. The system must enforce HTTPS by redirecting HTTP requests to HTTPS endpoints.

SSL certificates must be valid, properly configured, and automatically renewed before expiration.

PROTOCOL-2: WebSocket/SSE

Real-time features including chat streaming and progress updates shall use Server-Sent Events (SSE) over HTTP. SSE connections must maintain keep-alive with heartbeat messages every 30 seconds. The system must handle connection drops gracefully with automatic reconnection attempts using exponential backoff.

PROTOCOL-3: Database Protocol

Database connections shall use PostgreSQL wire protocol over TCP with SSL/TLS encryption enabled. Connection pooling must use persistent connections with maximum lifetime of 1 hour. The system must implement prepared statements for all parameterized queries to prevent SQL injection attacks.