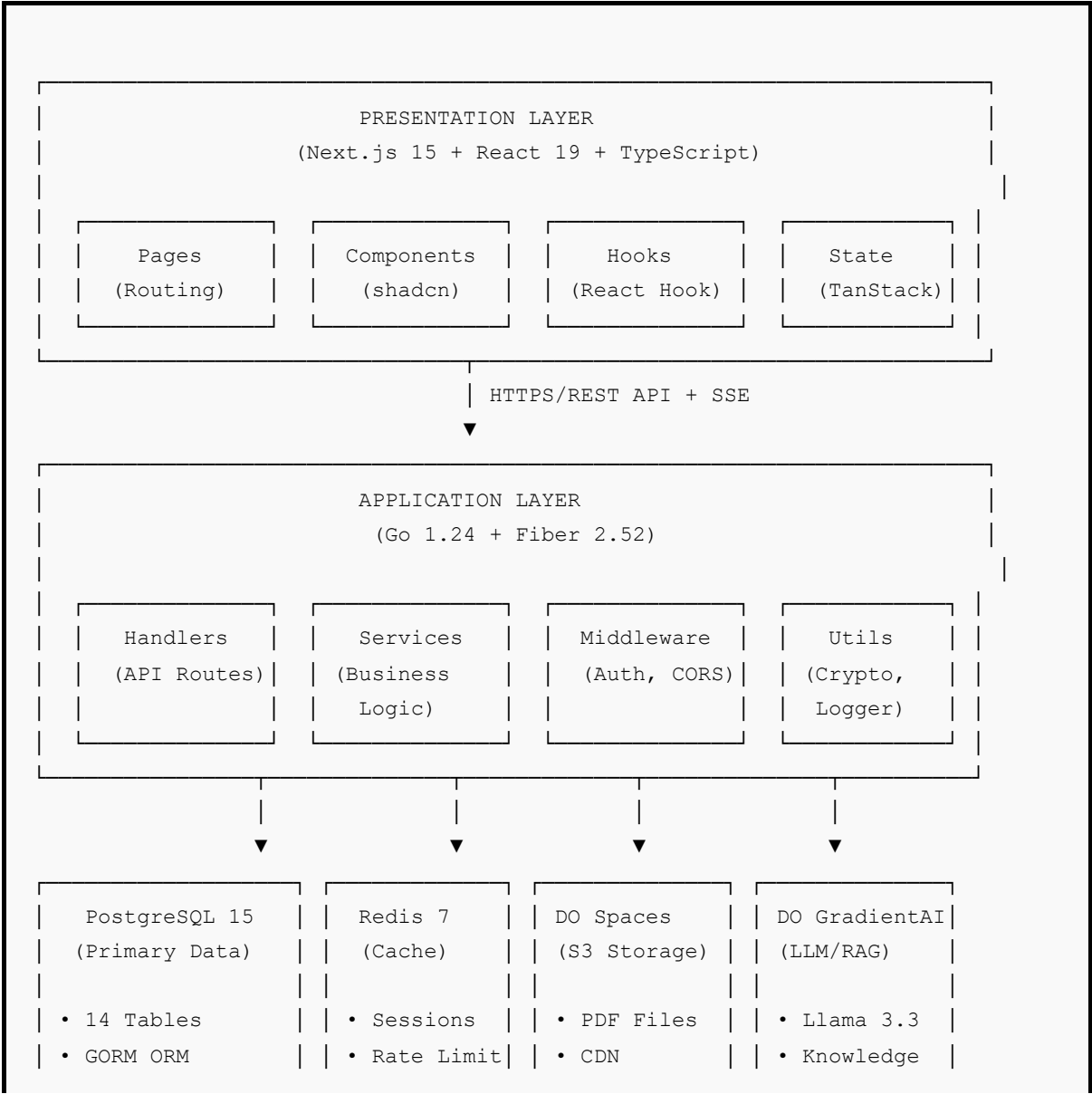


## 6. DESIGN

### 6.1 System Architecture

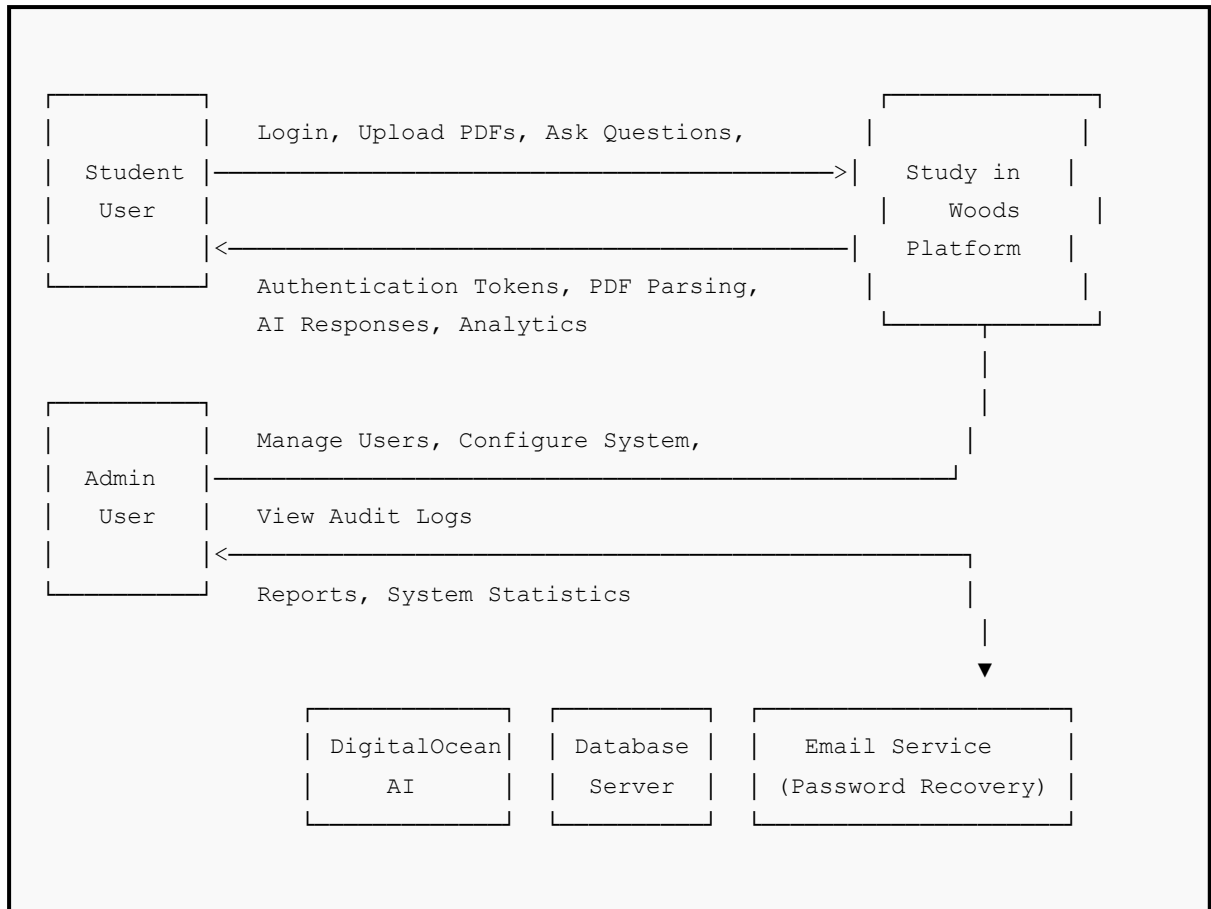
The Study in Woods platform implements a three-tier architecture consisting of Presentation Layer (Next.js frontend), Application Layer (Go Fiber backend API), and Data Layer (PostgreSQL database, Redis cache, DigitalOcean Spaces). This separation enables independent scaling, technology replacement, and clear responsibility boundaries. Communication between tiers occurs through well-defined RESTful APIs with JSON payloads, Server-Sent Events for real-time streaming, and S3-compatible protocols for file storage.



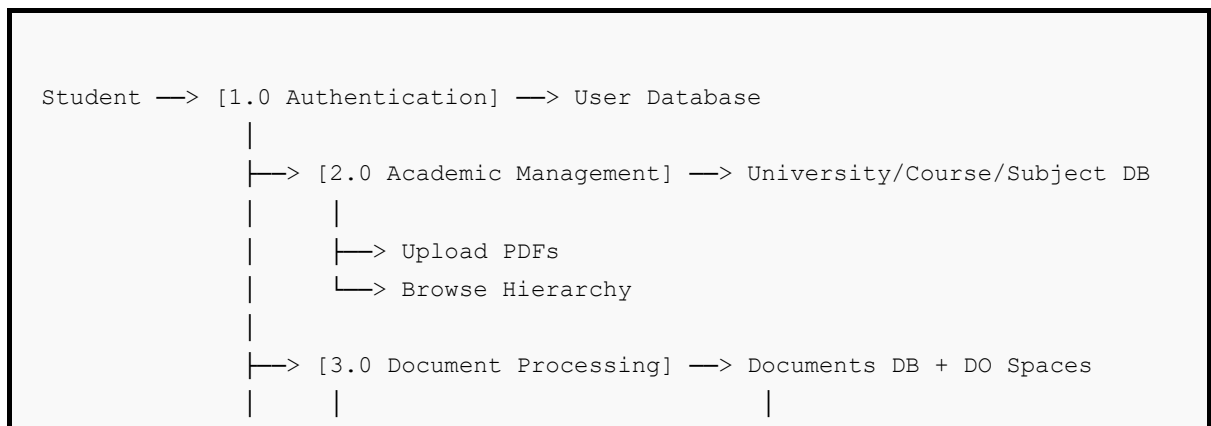
• Migrations	• Pub/Sub	• Pre-signed	Bases
--------------	-----------	--------------	-------

## 6.2 Data Flow Diagrams

### 6.2.1 Level 0 - Context Diagram



### 6.2.2 Level 1 - System Decomposition



```

    |
    |   └─> Upload PDF
    |   └─> Extract Syllabus ────┐─> AI Service
    |   └─> Index to Knowledge Base┘
    |
    |   └─> [4.0 AI Chat System] ──> ChatSessions + ChatMessages DB
    |   |
    |   |   └─> Create Session
    |   |   └─> Send Message ────┐─> AI Service + Knowledge Base
    |   |   └─> View History
    |   |
    |   └─> [5.0 Analytics] ──> Activity Logs + Usage Stats DB
    |
Admin ──> [6.0 Administration] ──> All Databases + Audit Logs
    |
    |   └─> User Management
    |   └─> System Configuration
    |   └─> View Audit Logs

```

### 6.2.3 Level 2 - Process Detail: Document Upload & Processing

```

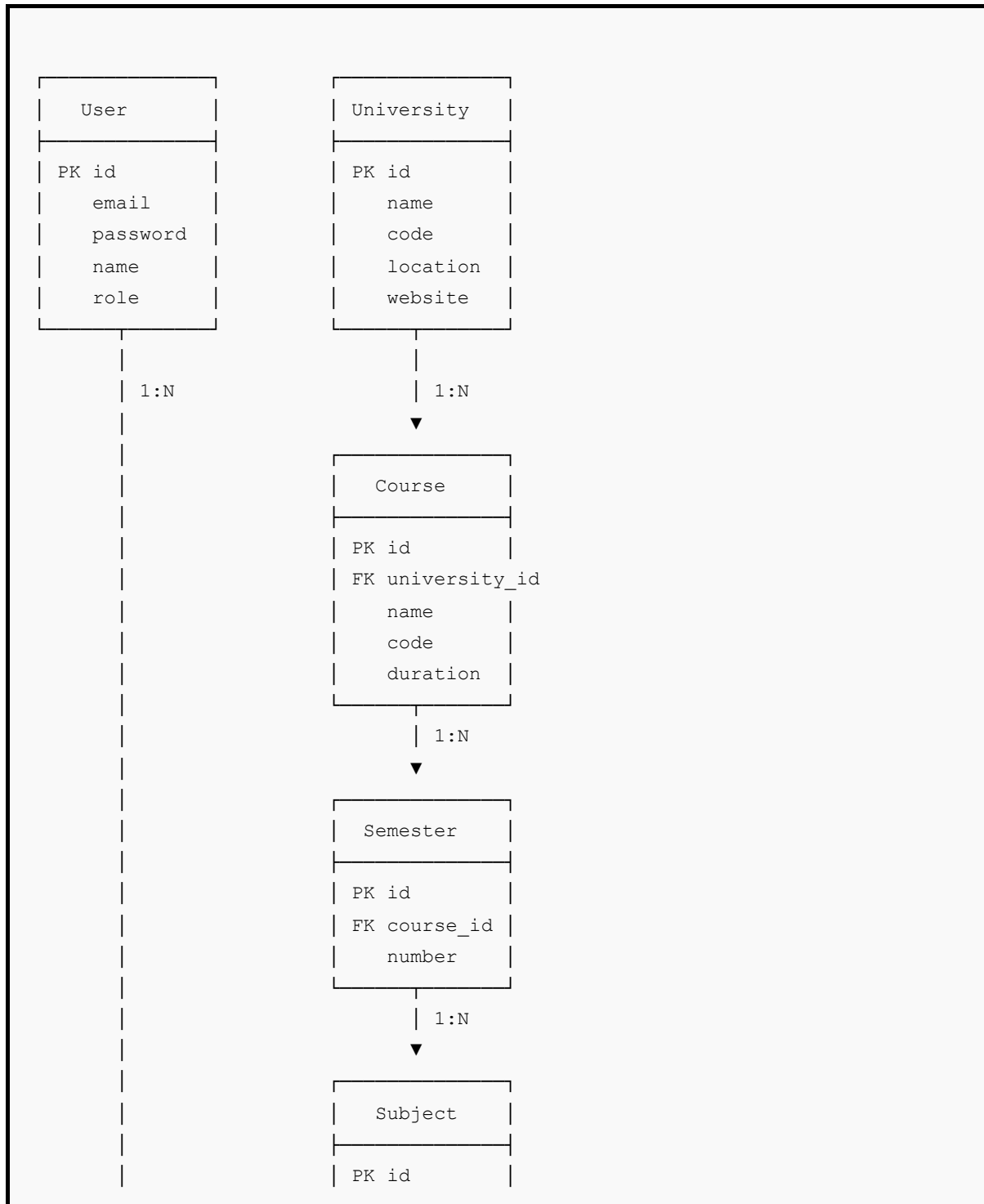
[3.0 Document Processing]
|
|   └─> [3.1 Validate Upload]
|   |   Input: File (from User)
|   |   Process: Check file type (PDF), size (<10MB), duplicate
|   |   Output: Validation Result
|   |
|   |   ▼
|   |   └─> [3.2 Upload to Storage]
|   |   |   Input: Validated File
|   |   |   Process: Generate UUID, Upload to DO Spaces (multipart if >5MB)
|   |   |   Output: Storage URL, Storage Key
|   |   |
|   |   |   ▼
|   |   |   └─> [3.3 Save Metadata]
|   |   |   |   Input: File metadata, Storage info
|   |   |   |   Process: Insert Document record with status='pending'
|   |   |   |   Output: Document ID
|   |   |   |   Database: Documents table
|   |   |   |
|   |   |   |   ▼
|   |   |   |   └─> [3.4 Extract Content]
|   |   |   |   |   Input: PDF URL
|   |   |   |   |   Process: Use AI to extract syllabus structure (units, topics)
|   |   |   |   |   AI Call: Llama 3.3 70B with structured prompt
|   |   |   |   |   Output: JSON { units: [ { title, topics: [...] } ] }
|   |   |   |
|   |   |   |   ▼

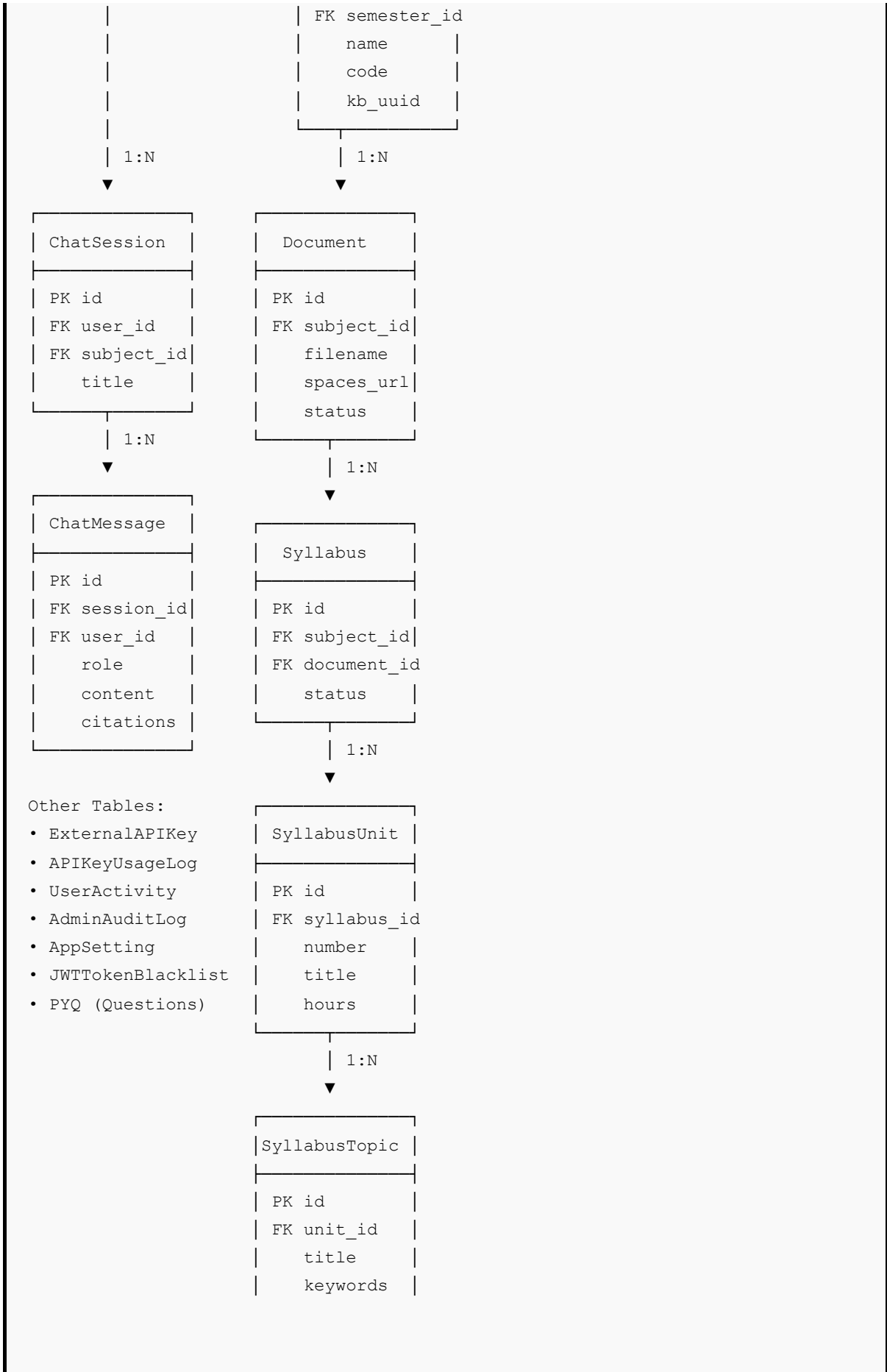
```

```
|> [3.5 Store Syllabus]
|   | Input: Extracted JSON
|   | Process: Parse and insert into Syllabus, SyllabusUnit,
SyllabusTopic
|   | Database: Syllabus tables (3 tables)
|   ▼
|> [3.6 Index to Knowledge Base]
|   | Input: Document ID, PDF URL
|   | Process: Upload to subject's Knowledge Base, Poll indexing status
|   | Update: Document.indexing_status = 'completed'
|   | External: DigitalOcean Knowledge Base API
```

### 6.3 Entity Relationship Diagram

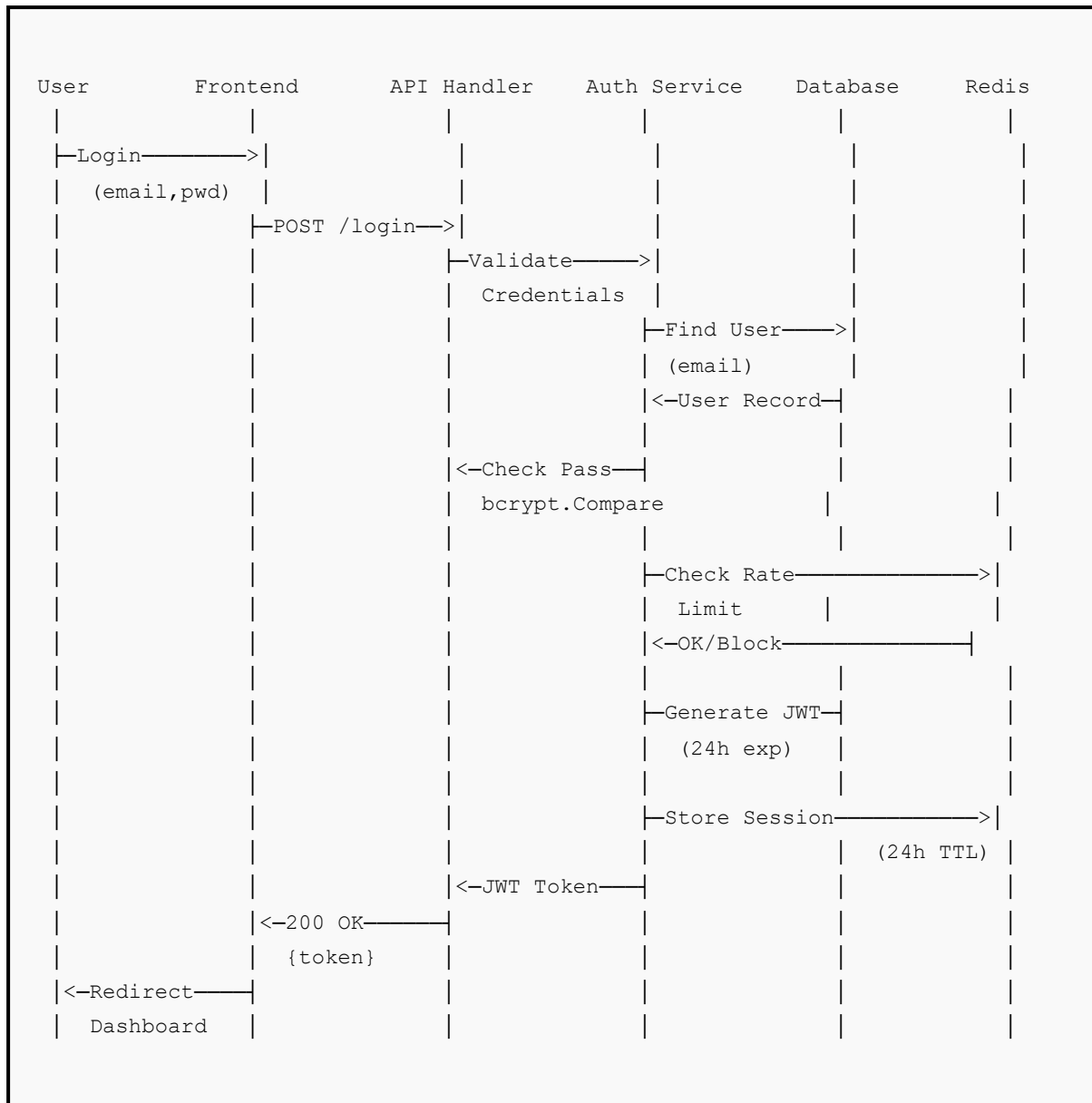
The database schema consists of 14 primary tables organized in a hierarchical structure. The core academic hierarchy flows from University → Course → Semester → Subject, with documents and chat sessions associated with subjects. User authentication and activity tracking tables support system security and analytics.





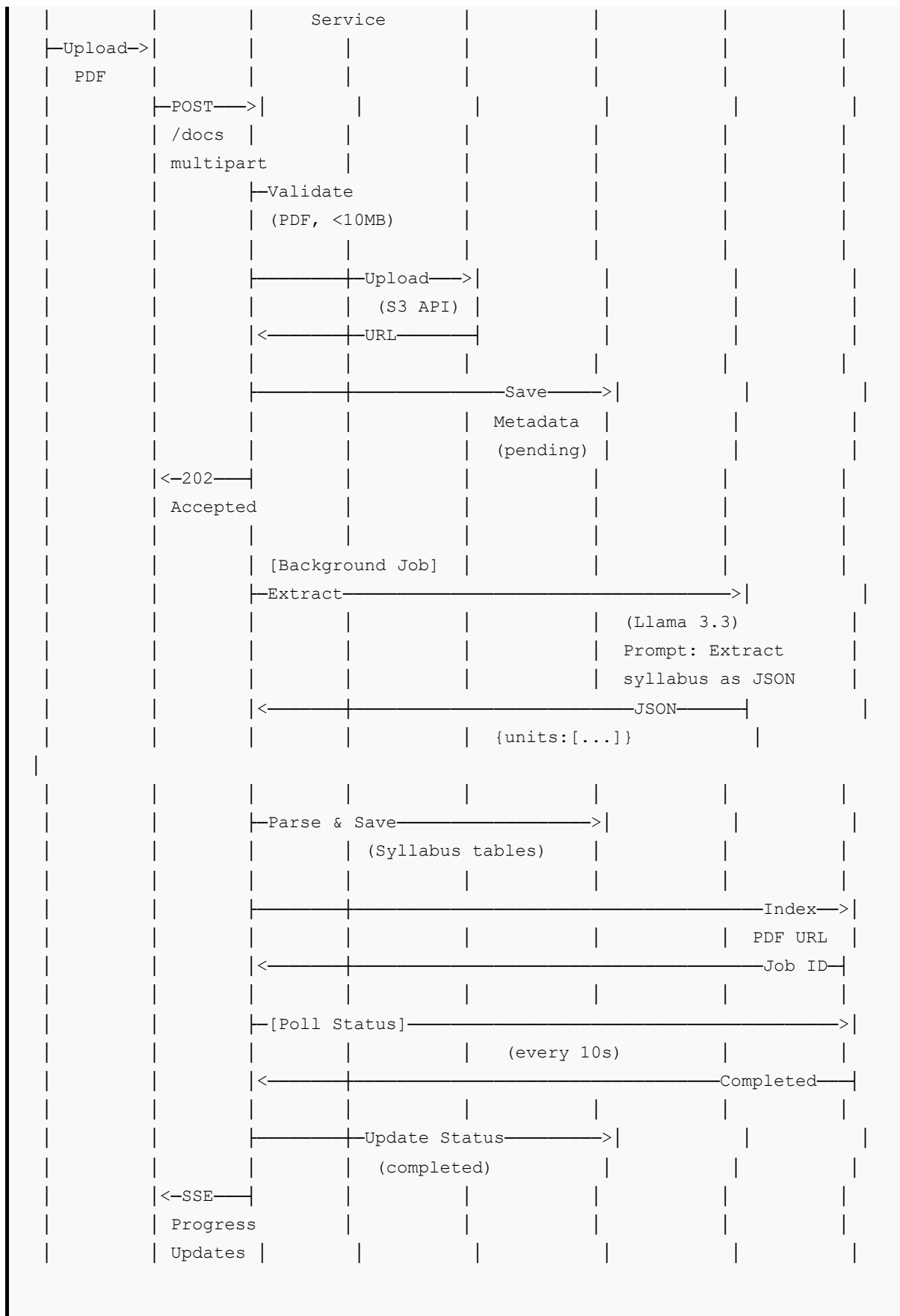
## 6.4 Sequence Diagrams

### 6.4.1 User Login Flow



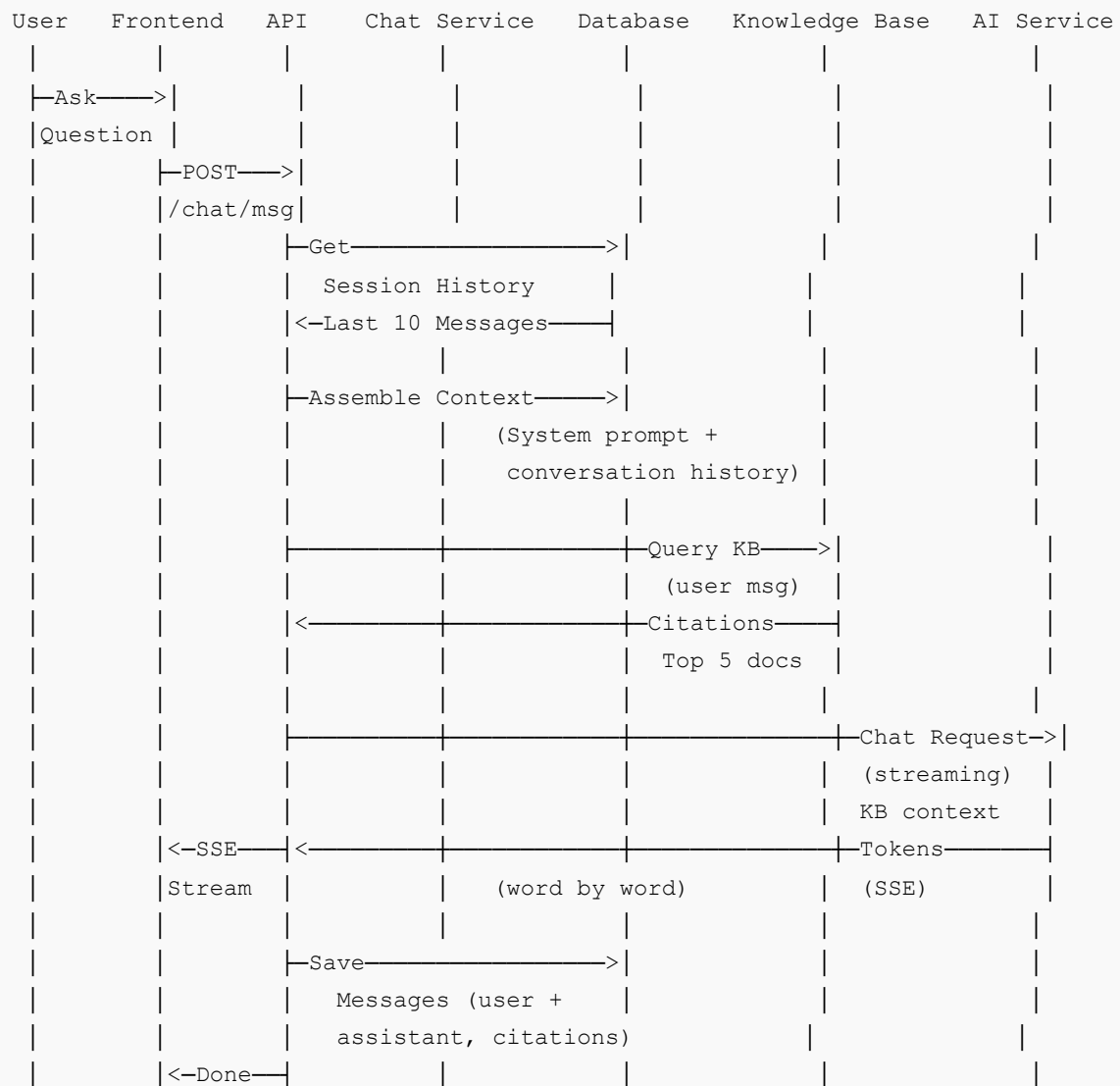
### 6.4.2 Document Upload & Syllabus Extraction Flow





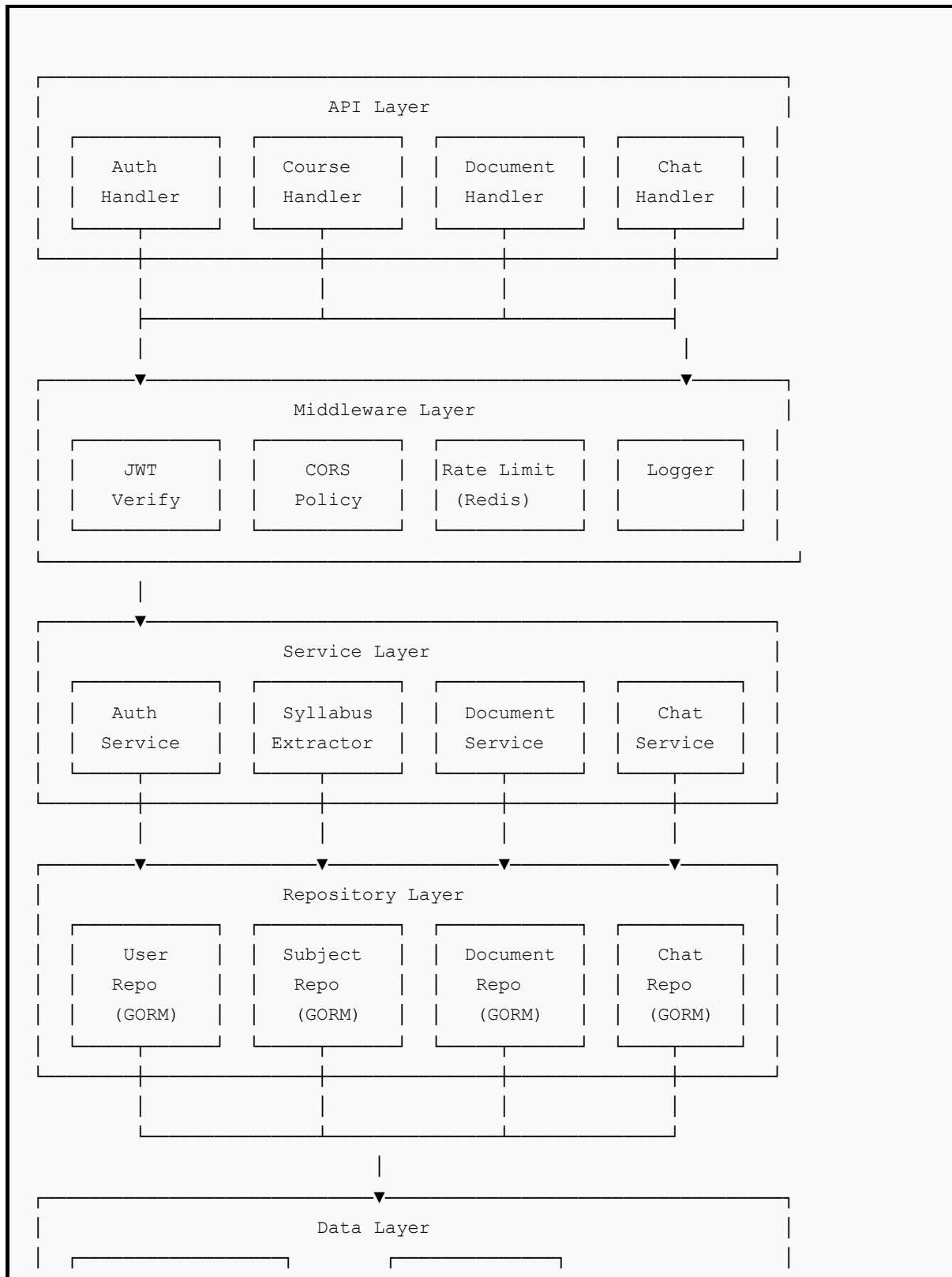
### 6.4.3 AI Chat Interaction Flow

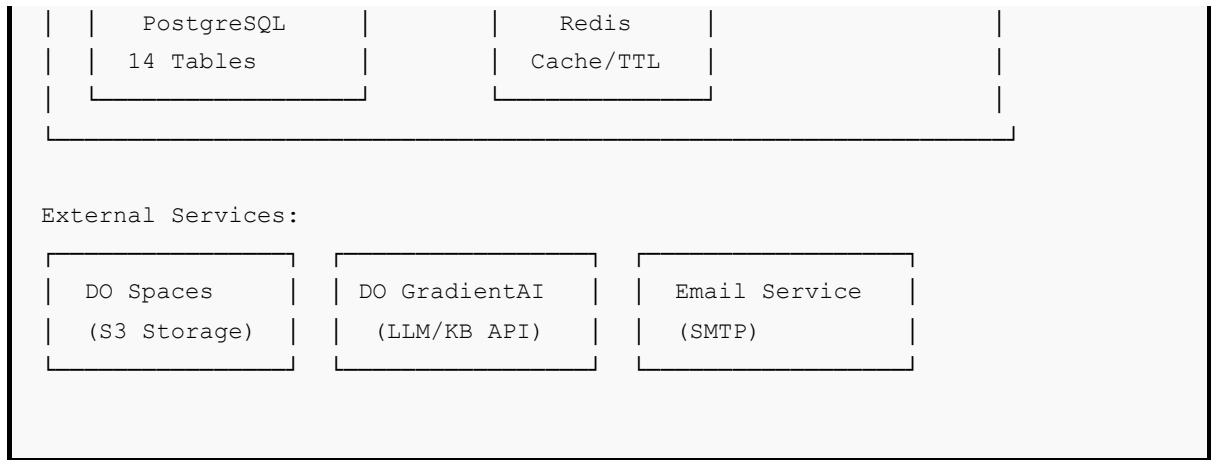




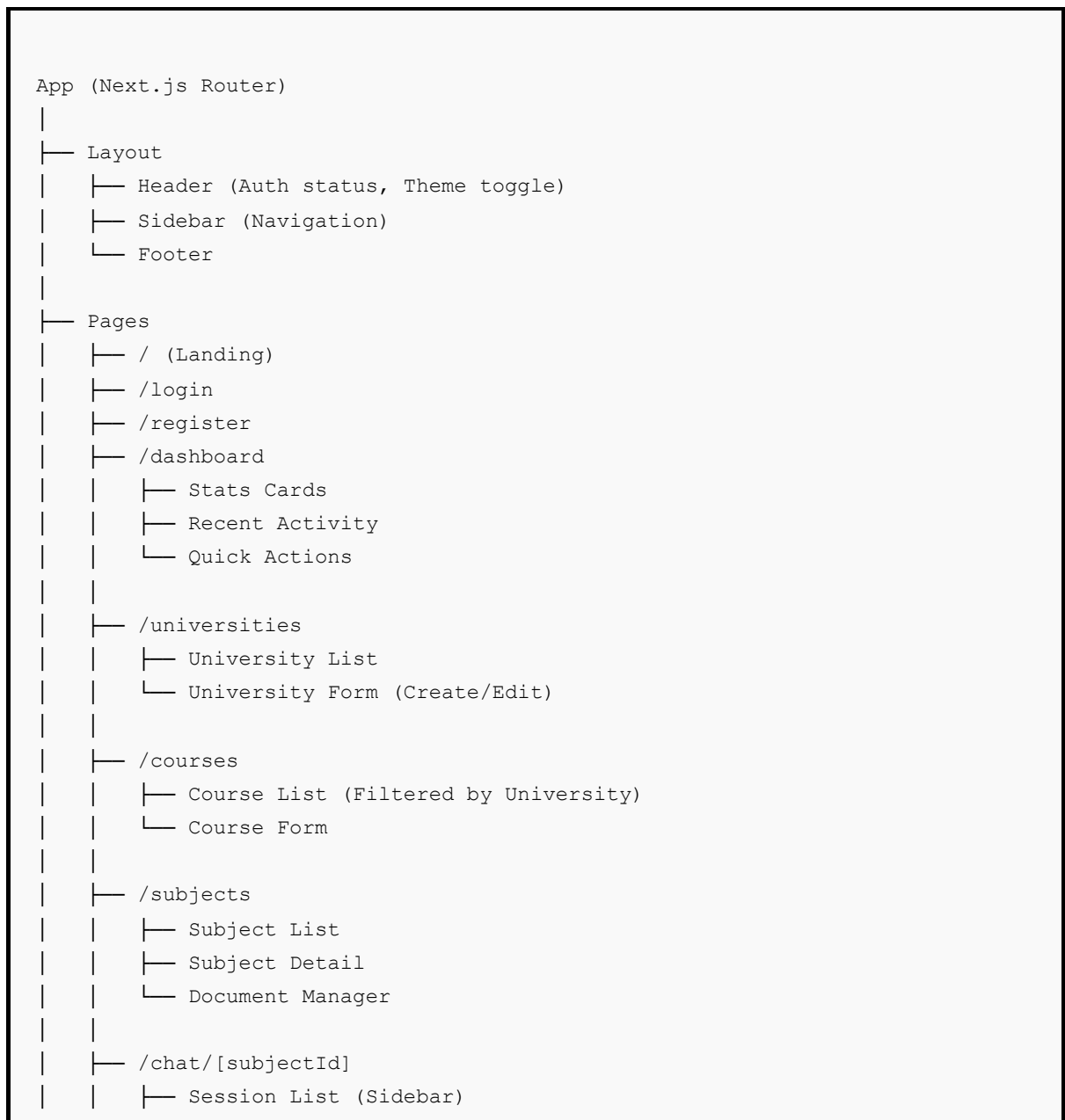
## 6.5 Component Diagrams

### 6.5.1 Backend Component Architecture





### 6.5.2 Frontend Component Hierarchy



```

| | | └─ Chat Area
| | |   └─ Message List
| | |     └─ User Message
| | |       └─ AI Message (with Citations)
| | |         └─ Input Box (with streaming)
| | |           └─ Citation Panel
| | |
| | └─ /analytics
| |   └─ Usage Charts (Recharts)
| |   └─ Statistics Cards
| |   └─ User Activity Table
| |
| └─ /admin
|   └─ User Management
|   └─ System Settings
|   └─ Audit Logs
|
└─ Providers
  └─ TanStack QueryProvider (API cache)
  └─ ThemeProvider (Dark mode)
  └─ AuthProvider (User context)
  └─ Utilities
    └─ API Client (Axios)
    └─ Hooks (useAuth, useChat, useUpload)
    └─ Validators (Zod schemas)

```