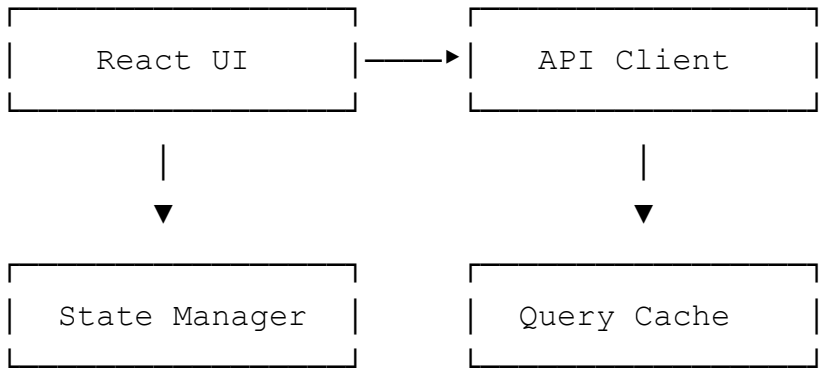
PDF Q&A Application - Technical Architecture

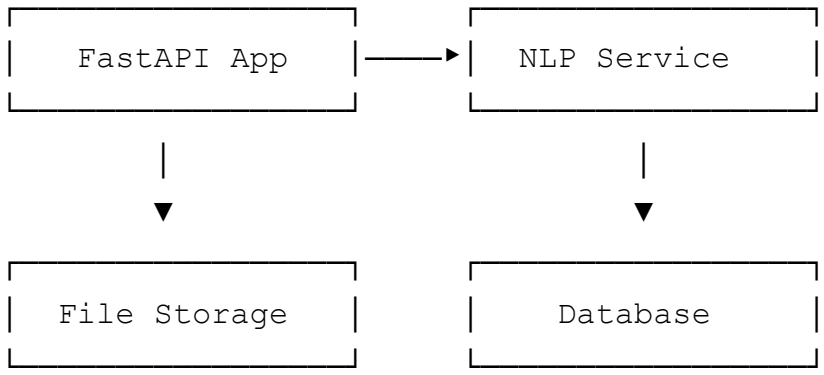
High-Level Design (HLD)

System Components

1. **Frontend Application**



2. **Backend Services**



Data Flow

1. **Document Upload Flow**

```
User → UI → API → File Storage → Database
                  └→ Document Processing
```

2. **Question-Answer Flow**

```
User → UI → API → NLP Service → Document Retrieval
                  └→ Answer Generation → UI
```

Low-Level Design (LLD)

Frontend Components

1. **UI Components**

```
// Component Hierarchy
App
├── Header
├── FileUpload
│   └── DropZone
├── DocumentList
│   └── DocumentCard
└── ChatInterface
    ├── MessageList
    │   └── MessageBubble
    └── MessageInput
```

2. **State Management**

```
// Zustand Store
interface Store {
  selectedDocument: Document | null;
  conversations: Record<string, Conversation>;
  // Actions
  selectDocument: (doc: Document) => void;
  addMessage: (docId: string, msg: Message) => void;
}
```

3. **API Integration**

```
// API Client
class ApiClient {
  uploadDocument(file: File): Promise<Document>;
  askQuestion(docId: string, question: string): Promise<Answer>;
}
```

Backend Components

1. **API Routes**

```
# FastAPI Routes
@app.post("/api/upload")
async def upload_document(file: UploadFile)

@app.post("/api/question")
async def ask_question(request: QuestionRequest)
```

2. **Service Layer**

```
# Document Service
class DocumentService:
    async def save_document(file: UploadFile)
    def get_document(id: str)

# QA Service
class QAService:
    async def get_answer(doc_id: str, question: str)
```

3. **Database Models**

```
# SQLAlchemy Models
class Document(Base):
    id: str
    name: str
    path: str
    upload_date: datetime
    size: int
```

Security Architecture

1. **Input Validation**

```
# Request Models
class QuestionRequest(BaseModel):
    document_id: str
    question: str
```

2. **File Validation**

```
def validate_pdf(file: UploadFile):
    if not file.filename.endswith('.pdf'):
        raise HTTPException(400)
```

Error Handling

1. **Frontend Error Handling**

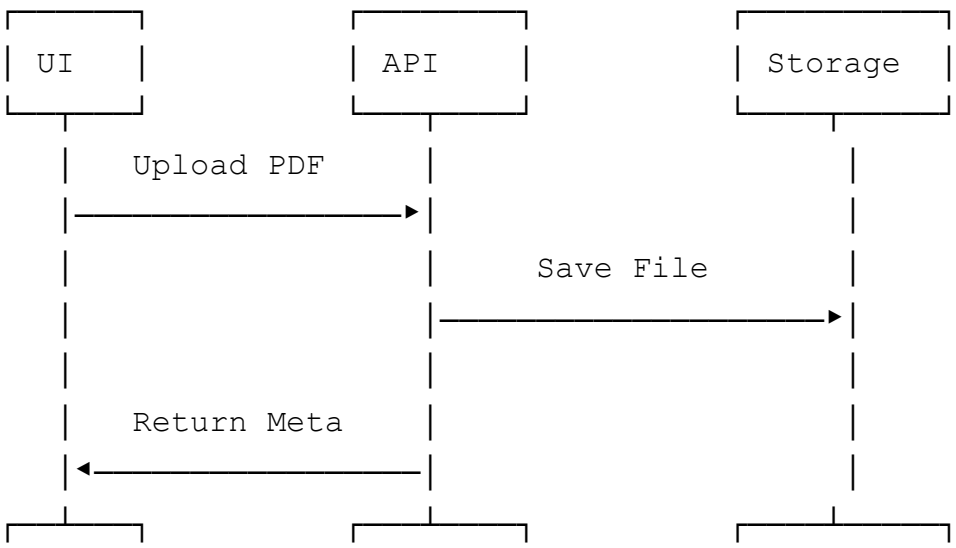
```
try {
  await api.uploadDocument(file);
} catch (error) {
  handleError(error);
}
```

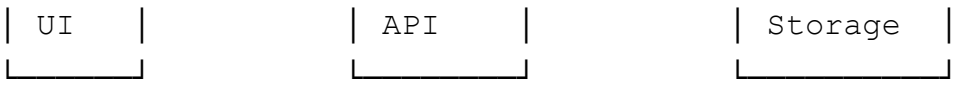
2. **Backend Error Handling**

```
@app.exception_handler(HTTPException)
async def http_exception_handler(request, exc):
    return JSONResponse(
        status_code=exc.status_code,
        content={"detail": exc.detail}
    )
```

System Interactions

Document Upload Sequence





Question-Answer Sequence