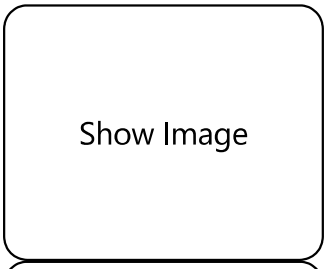
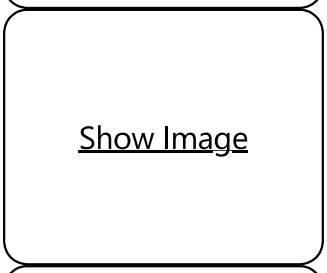
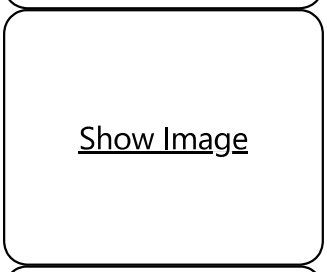
🧠 NeuroPDF — Your Intelligent Reading & Research Companion

> Transform PDFs into interactive knowledge networks that adapt to your persona and goals







🎯 Project Overview

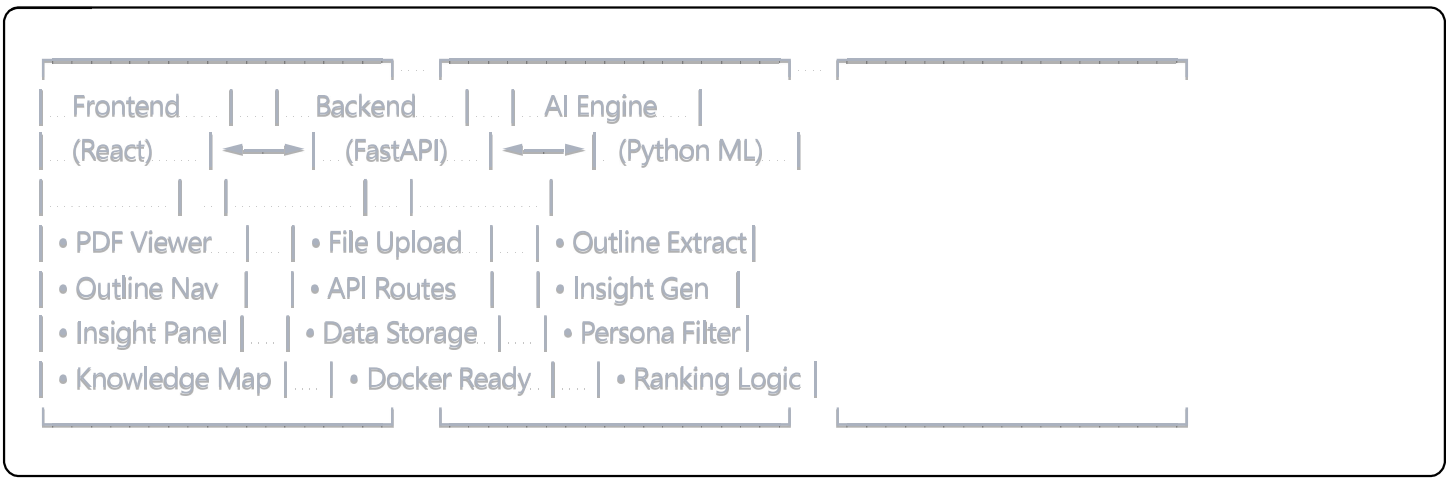
NeuroPDF is an AI-powered PDF analysis tool that extracts document structure, generates persona-based insights, and presents them through an interactive web interface. Built for researchers, students, and professionals who need to quickly understand and navigate complex documents.

🔥 Key Features

- 🧭 **Smart Document Structure Extraction** - Automatically identifies headings, sections, and page numbers
- 👤 **Persona-Driven Insights** - AI generates relevant highlights based on your role (Student, Researcher, Analyst)
- 📄 **Interactive PDF Viewer** - Powered by Adobe PDF Embed API with seamless navigation

- 🧠 **Knowledge Mapping** - Visual representation of document concepts and relationships
- ⚡ **Offline Processing** - Dockerized backend for complete offline usage

🧬 Architecture



🚀 Quick Start

Prerequisites

- **Node.js** 18+ and npm
- **Python** 3.9+
- **Docker** (optional, for containerized backend)
- **Git**

1. Clone Repository

```bash
git clone https://github.com/yourusername/neuropdf.git
cd neuropdf
```

2. Backend Setup (Round 1 - AI Processing)

```bash
```

```
# Navigate to backend
cd backend

# Create virtual environment
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt

# Run FastAPI server
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

Backend will be available at: `http://localhost:8000`

3. Frontend Setup (Round 2 - Web Interface)

```bash
# Navigate to frontend (new terminal)
cd frontend

# Install dependencies
npm install

# Start development server
npm run dev
```

Frontend will be available at: `http://localhost:5173`

4. Docker Setup (Alternative)

```bash
# Build and run backend container
cd backend
docker build -t neuropdf-backend .
docker run -p 8000:8000 neuropdf-backend

# Build and run frontend container
cd ../frontend
docker build -t neuropdf-frontend .
docker run -p 5173:5173 neuropdf-frontend
```

📁 Project Structure

```
neuropdf/
├── 📁 backend/                # Round 1: AI Processing Engine
│   ├── app/
│   │   ├── main.py            # FastAPI application entry
│   │   ├── routes/
│   │   │   ├── pdf_routes.py  # PDF upload & processing endpoints
│   │   │   └── data_routes.py # Data serving endpoints
│   │   ├── services/
│   │   │   ├── outline_extractor.py   # Document structure extraction
│   │   │   └── insight_generator.py   # Persona-based insight generation
│   │   ├── utils/
│   │   │   ├── file_handler.py        # File operations
│   │   │   └── json_response.py       # Response formatting
│   │   └── data/
│   │       ├── uploads/               # Uploaded PDFs
│   │       ├── outline.json           # Extracted document structure
│   │       └── insights.json          # Generated insights
│   ├── Dockerfile
│   ├── requirements.txt
│   └── README.md
├── 📁 frontend/               # Round 2: Web Interface
│   ├── src/
│   │   ├── components/
│   │   │   ├── PDFViewer.jsx          # Adobe PDF Embed integration
│   │   │   ├── OutlineSidebar.jsx     # Document navigation
│   │   │   ├── InsightsPanel.jsx      # AI-generated highlights
│   │   │   ├── PersonaSelector.jsx    # Role-based filtering
│   │   │   └── KnowledgeMap.jsx       # Visual concept mapping
│   │   ├── hooks/
│   │   │   └── useAPI.js              # Backend API integration
│   │   ├── utils/
│   │   │   └── constants.js           # App configuration
│   │   ├── App.jsx
│   │   └── main.jsx
│   ├── package.json
│   ├── vite.config.js
│   └── README.md
├── 📁 docker/
│   ├── docker-compose.yml             # Multi-container setup
│   └── .dockerignore
├── 📁 docs/
│   ├── api-documentation.md
│   ├── architecture-diagram.png
```

```
│   └── user-guide.md
├── 📁 sample-data/
│   ├── sample.pdf              # Test document
│   ├── outline.json            # Sample structure output
│   └── insights.json           # Sample insights output
└── README.md                   # This file
```

🧠 Round 1: AI Processing Engine

Outline Extractor (Round 1A)

Extracts document structure using font size heuristics and layout analysis:

```python
# Example output structure
{
  "title": "Understanding Artificial Intelligence",
  "outline": [
    {"level": "H1", "text": "Introduction", "page": 1},
    {"level": "H2", "text": "History of AI", "page": 2},
    {"level": "H3", "text": "Early Developments", "page": 3}
  ]
}
```

Insight Generator (Round 1B)

Generates persona-specific insights using semantic analysis:

```python
```

```
# Example persona-based output
{
    "persona": "Student",
    "goal": "Exam Preparation",
    "sections": [
        {
            "document": "ai-textbook.pdf",
            "section_title": "Neural Networks",
            "refined_text": "Key concepts for exam...",
            "page_number": 15,
            "importance_rank": 1,
            "confidence_score": 0.95
        }
    ]
}
```

💻 Round 2: Web Interface

Core Components

- **PDFViewer**: Adobe PDF Embed API integration with programmatic navigation
- **OutlineSidebar**: Hierarchical document structure with click-to-jump functionality
- **InsightsPanel**: Ranked, persona-filtered content highlights
- **PersonaSelector**: Dynamic role switching (Student, Researcher, Analyst, Professional)
- **KnowledgeMap**: Interactive concept visualization using React Flow

API Integration

The frontend communicates with the backend through these endpoints:

Method	Endpoint	Description
`POST`	`/upload`	Upload and process PDF documents
`GET`	`/outline`	Retrieve document structure
`GET`	`/insights`	Get persona-based insights
`POST`	`/persona`	Update persona and regenerate insights
`GET`	`/health`	Backend health check

🎭 Persona System

Available Personas

- 🎓 **Student**: Focus on key concepts, definitions, and exam-relevant content
- 🔬 **Researcher**: Emphasize methodology, citations, and novel findings
- 📊 **Analyst**: Highlight data, statistics, and analytical frameworks
- 💼 **Professional**: Prioritize practical applications and business implications

Usage Example

```javascript
// Switch persona and get updated insights
const response = await fetch('/api/persona', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
        persona: 'Student',
        goal: 'Exam Preparation'
    })
});
```

🐳 Docker Deployment

Single Container

```bash
# Backend only
docker run -p 8000:8000 neuropdf-backend

# Frontend only
docker run -p 5173:5173 neuropdf-frontend
```

Multi-Container with Docker Compose

```bash
docker-compose up -d
```

This starts both backend (port 8000) and frontend (port 5173) services.

🧪 Testing

Backend Tests

```bash
cd backend
pytest tests/ -v
```

Frontend Tests

```bash
cd frontend
npm test
```

Integration Tests

```bash
# Test full workflow with sample PDF
python scripts/test_pipeline.py
```

📊 Sample User Flow

For Students (Exam Preparation)

1. **Upload**: Drop PDF textbook into the interface
2. **Process**: Backend extracts structure and generates student-focused insights
3. **Navigate**: Use outline sidebar to jump between chapters
4. **Study**: Review AI-highlighted key concepts in insights panel
5. **Visualize**: Explore knowledge map to understand topic relationships

For Researchers (Literature Review)

1. **Batch Upload**: Add multiple research papers
2. **Analyze**: Generate researcher-focused insights emphasizing methodology
3. **Compare**: Cross-reference findings across documents
4. **Export**: Save curated insights for citation management

🔧 Configuration

Backend Configuration

Edit `backend/app/config.py`:

```python
# AI Model Settings
EMBEDDING_MODEL = "sentence-transformers/all-MiniLM-L6-v2"
MAX_FILE_SIZE = 50 * 1024 * 1024  # 50MB
SUPPORTED_FORMATS = [".pdf"]

# Processing Settings
MIN_SECTION_LENGTH = 100
INSIGHT_RANKING_THRESHOLD = 0.7
```

Frontend Configuration

Edit `frontend/src/utils/constants.js`:

```javascript
export const API_BASE_URL = process.env.VITE_API_URL || 'http://localhost:8000';
export const ADOBE_CLIENT_ID = process.env.VITE_ADOBE_CLIENT_ID;
export const MAX_UPLOAD_SIZE = 50 * 1024 * 1024; // 50MB
```

🚧 Development Roadmap

Phase 1 (Current)

- ✅ PDF structure extraction
- ✅ Basic persona system
- ✅ React web interface
- ✅ Docker containerization

Phase 2 (Future)

- 🔄 Multi-PDF cross-referencing
- 🔄 Advanced knowledge graph
- 🔄 Real-time collaboration
- 🔄 Mobile application

Phase 3 (Advanced)

- 📋 Integration with reference managers

- 📋 AI-powered note generation
- 📋 Advanced search capabilities
- 📋 Cloud deployment options

🤝 Contributing

1. Fork the repository
2. Create feature branch (`git checkout -b feature/amazing-feature`)
3. Commit changes (`git commit -m 'Add amazing feature'`)
4. Push to branch (`git push origin feature/amazing-feature`)
5. Open Pull Request

Development Guidelines

- Follow PEP 8 for Python code
- Use ESLint configuration for JavaScript
- Write tests for new features
- Update documentation

📋 API Documentation

Complete API documentation is available at `http://localhost:8000/docs` when running the backend server.

Key Endpoints

bash

```
# Upload and process PDF
curl -X POST "http://localhost:8000/upload" \
    -H "Content-Type: multipart/form-data" \
    -F "file=@sample.pdf"

# Get document outline
curl "http://localhost:8000/outline"

# Get persona-based insights
curl "http://localhost:8000/insights"

# Update persona
curl -X POST "http://localhost:8000/persona" \
    -H "Content-Type: application/json" \
    -d '{"persona": "Student", "goal": "Exam Prep"}'
```

🔍 Troubleshooting

Common Issues

Backend not starting:

bash

```
# Check Python version
python --version  # Should be 3.9+

# Reinstall dependencies
pip install -r requirements.txt --force-reinstall
```

Frontend build errors:

bash

```
# Clear node modules and reinstall
rm -rf node_modules package-lock.json
npm install
```

Docker issues:

bash

```
# Rebuild containers
docker-compose down
docker-compose build --no-cache
docker-compose up
```

Performance Optimization

- Use SSD storage for faster PDF processing
- Allocate sufficient RAM (minimum 4GB recommended)
- Enable GPU acceleration for large document processing (optional)

📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

🙏 Acknowledgments

- Adobe PDF Embed API for document viewing
- Sentence Transformers for semantic analysis
- FastAPI for backend framework
- React Flow for knowledge visualization
- The open-source community for various tools and libraries

📞 Support

- **Documentation**: docs/
- **Issues**: GitHub Issues
- **Discussions**: GitHub Discussions

Made with ❤️ for better document understanding