PDF Slice & Dice

> Professional PDF extraction and conversion tool with AI-powered insights. Process PDFs entirely in your browser - no server required for core features.

![CI Tests](https://github.com/sjdevpl/Pdfslicendice/actions/workflows/ci.yml)
![Deploy to GitHub Pages](https://github.com/sjdevpl/Pdfslicendice/actions/workflows/deploy.yml)
![GitHub Pages](https://sjdevpl.github.io/Pdfslicendice/)

Try it live on GitHub Pages | Full version with AI features

Features

Core Features (Available on GitHub Pages)

- PDF Upload & Preview: Upload and view PDF files in your browser
- Page Extraction: Select and extract individual pages from PDFs
- Batch Operations: Process multiple pages at once
- Format Conversion: Export pages to multiple formats:

- PDF
- Image (PNG)
- Word Document (DOCX)
- PowerPoint Presentation (PPTX)

- 100% Client-Side: All PDF processing runs in your browser - no data leaves your device
- Offline Support: Works without internet connection once loaded

AI Features (Full Version Only)

- AI-Powered Analysis: Get intelligent summaries and insights from PDF pages
- Keyword Extraction: Automatically extract important keywords
- Batch AI Processing: Analyze multiple pages at once

Quick Start

Try Online

Visit the live demo: https://sjdevpl.github.io/Pdfslicendice/

Run Locally

Prerequisites: Node.js (v20 or higher)

1. Clone the repository

```
`bash
git clone https://github.com/sjdevpl/Pdfslicendice.git
cd Pdfslicendice
`
```

2. Install dependencies

```
`bash
npm install
`
```

3. Run the development server

```
`bash
npm run dev
`
```

The app will be available at http://localhost:5173

> Note: For AI features, you'll need to set up the backend server (see Full Setup below).

Tech Stack

- Frontend Framework: React 19 + TypeScript
- Build Tool: Vite
- Styling: Tailwind CSS
- PDF Processing:

- pdf-lib - PDF manipulation
- pdfjs-dist - PDF rendering

- Document Generation:

- docx - Word document creation
- pptxgenjs - PowerPoint generation

- AI Integration: Google Gemini API (via backend)
- Testing: Vitest + React Testing Library
- CI/CD: GitHub Actions

Available Scripts

Development

```
npm run dev            # Start development server
npm run server:dev     # Start backend server with hot reload
```

Building

```
npm run build          # Build for production
npm run preview        # Preview production build
```

Testing

```
npm test               # Run tests in watch mode
npm run test:run       # Run tests once
npm run test:ui        # Run tests with UI
npm run test:coverage  # Run tests with coverage
```

Production

```
npm run server         # Start production backend server
```

Full Setup (with AI Features)

To enable AI-powered features, you need to set up the backend server:

1. Configure environment variables

`bash

```
cp .env.local.template .env.local
```

`

2. Edit .env.local and add:

`env

```
GEMINI_API_KEY=your_gemini_api_key_here
VITE_BACKEND_URL=http://localhost:3001
```

`

Get your Gemini API key from: https://aistudio.google.com/app/apikey

3. Start the backend server (in one terminal):

`bash

```
npm run server:dev
```

`

The backend will run on http://localhost:3001

4. Start the frontend (in another terminal):
`bash
npm run dev
`

Architecture

Frontend (Client-side)

- PDF loading and rendering using PDF.js
- Page extraction and splitting
- Format conversion (PDF Image, Word, PowerPoint)
- Works completely offline once loaded
- No server required for core features

Backend (Server-side) - Optional

- Gemini API integration
- AI-powered content analysis
- Secure API key handling
- Only required for AI features

Security

The Gemini API key is stored only on the backend server and never exposed to the frontend. This ensures:

- API key never appears in browser code
- Prevents unauthorized API usage
- Protects against key theft from client-side code
- All AI requests go through your secure backend

Testing

This project includes a comprehensive test suite:

```
npm test              # Run tests in watch mode
npm run test:run        # Run tests once
npm run test:ui         # Run tests with interactive UI
npm run test:coverage    # Generate coverage report
```

Tests are automatically run on every push and pull request via GitHub Actions CI.

See tests/README.md for more details.

Building for Production

Build the production version:

```
npm run build
```

The output will be in the dist/ folder. All core PDF features work offline without requiring the backend.

Deploying to GitHub Pages

The project is automatically deployed to GitHub Pages on every push to main. See GITHUB_PAGES_SETUP.md for setup instructions.

Contributing

Contributions are welcome! Please feel free to submit a Pull Request. For major changes, please open an issue first to discuss what you would like to change.

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

License

This project is open source and available under the MIT License (or your preferred license).

Links

- Live Demo (GitHub Pages): https://sjdevpl.github.io/Pdfslicendice/
- Full Version: https://pdfslicendice.sjdev.pl
- Repository: https://github.com/sjdevpl/Pdfslicendice

Acknowledgments

- Built with React and Vite
- PDF processing powered by pdf-lib and PDF.js
- AI features powered by Google Gemini

Made with by sjdevpl