HighlightTool.tsx

```
import { Highlighter } from "lucide-react";
const HighlightTool = () => {
const handleHighlight = async () => {
alert("Highlight tool activated!");
// Capture the highlight data (you can modify this to capture actual highlights)
const highlightData = "Highlighted text or area";
// Send the highlight data to the backend
const response = await fetch("http://127.0.0.1:5000/save-highlight/1", { // 1 is the PDF ID
method: "POST",
headers: {
"Content-Type": "application/json",
},
body: JSON.stringify({
highlightData,
}),
});
if (response.ok) {
const data = await response.json();
console.log("Highlight saved successfully:", data);
} else {
console.error("Failed to save highlight:", response.statusText);
}
};
return (
<button onClick={handleHighlight} className="flex items-center bg-yellow-500 text-white
px-4 py-2 rounded-lg">
<Highlighter className="w-5 h-5 mr-2" />
Highlight Tool
</button>
);
};
export default HighlightTool;
```

Navbar.tsx

```
import React, { useState } from "react";

const Navbar = () => {
  const [selectedFile, setSelectedFile] = useState<File | null>(null);

  const handleFileUpload = (event: React.ChangeEvent<HTMLInputElement>) => {
    const file = event.target.files?.[0];
    if (file) {
      setSelectedFile(file);
      console.log("File uploaded:", file.name); // ✅ Debugging check
    }
  };
  return (
    <nav className="w-full fixed top-0 left-0  p-4 shadow-md">
      <input id="file-input" type="file" onChange={handleFileUpload} className="hidden" />
      {selectedFile && <p className="text-blue-500 mt-2">Selected File:
{selectedFile.name}</p>}
    </nav>
  );
};
export default Navbar;
```

PDFList.tsx

```
import { useState, useEffect } from "react";

const PDFList = () => {
  const [pdfs, setPdfs] = useState<{ id: number; filename: string; folder: string }[]>([]);

  useEffect(() => {
    const fetchPDFs = async () => {
      try {
        const response = await fetch("http://127.0.0.1:5000/get-pdfs");
        if (!response.ok) {
          throw new Error(`Failed to fetch PDFs: ${response.statusText}`);
        }
        const data = await response.json();
        setPdfs(data);
      } catch (error) {
        console.error("Error fetching PDFs:", error);
      }
    };

    fetchPDFs();
  }, []); // Dependency array left empty to fetch data only once on mount

  return (
    <div className="p-4 border rounded-lg">
      <h2 className="text-lg font-bold mb-2">Uploaded PDFs</h2>
      <ul>
        {pdfs.map((pdf) => (
          <li key={pdf.id}>
            <a
              href={`http://127.0.0.1:5000/uploads/${pdf.folder}/${pdf.filename}`} // Correctly using
template literals
              target="_blank"
              rel="noopener noreferrer"
              className="text-blue-500 hover:underline"
            >
              {pdf.filename}
            </a>
          </li>
        ))}
      </ul>
    </div>
  );
};
export default PDFList;
```

PDFViewer.tsx

```
import React, { useState, useEffect } from "react";
import { Worker, Viewer } from "@react-pdf-viewer/core";
import "@react-pdf-viewer/core/lib/styles/index.css";
import { Upload, FolderOpen } from "lucide-react";
import SnippingTool from "./SnippingTool";

interface PDFViewerProps {
  pdfList: { filename: string; url: string }[];
  selectedFolder: string | null;
}

const PDFViewer: React.FC<PDFViewerProps> = ({ pdfList, selectedFolder }) => {
  const [selectedPdf, setSelectedPdf] = useState<string | null>(null);

  useEffect(() => {
    if (pdfList.length > 0 && !selectedPdf) {
      setSelectedPdf(pdfList[0].url); // ✅ Fix PDF auto-selection
    }
  }, [pdfList]);

  return (
    <div className="flex h-screen w-full bg-gray-900 text-white overflow-hidden">
      {/* Sidebar */}
      <div className="w-1/4 p-4 border-r border-gray-700">
        <h2 className="text-lg font-bold mb-4 flex items-center gap-2">
          <FolderOpen className="w-5 h-5 text-blue-400" /> My Documents
        </h2>
        <SnippingTool selectedFolder={selectedFolder} /> {/* ✅ Pass selectedFolder */}
      </div>

      {/* Main Content */}
      <div className="flex flex-col flex-grow p-6">
        <div className="flex flex-col items-center w-full">
          {/* File Upload Section */}
          <div className="w-full flex items-center justify-center mb-4 gap-4">
```

```
            <input
              type="file"
              className="border border-gray-600 rounded-lg bg-gray-800 text-white p-2
cursor-pointer"
            />
            <button className="flex items-center gap-2 px-4 py-2 bg-blue-600
hover:bg-blue-700 text-white rounded-lg">
              <Upload className="w-4 h-4" /> Upload PDF
            </button>
          </div>

          {/* PDF Selection */}
          {pdfList.length === 0 ? (
            <p className="text-center text-gray-400">No PDFs available</p>
          ) : (
            <>
              <select
                className="w-full p-3 border border-gray-600 rounded-lg mb-4 bg-gray-800
text-white"
                onChange={(e) => setSelectedPdf(e.target.value)}
                value={selectedPdf || ""}
              >
                <option value="" disabled>Select a PDF</option>
                {pdfList.map((pdf, index) => (
                  <option key={index} value={pdf.url}>
                    {pdf.filename}
                  </option>
                ))}
              </select>

              {/* PDF Viewer */}
              {selectedPdf ? (
                <div className="w-full flex-grow border border-gray-600 p-4 rounded-lg
bg-gray-800 shadow-lg overflow-auto h-[80vh]">
                  <Worker
workerUrl="https://unpkg.com/pdfjs-dist@3.11.174/build/pdf.worker.min.js">
                    <Viewer fileUrl={selectedPdf} />
                  </Worker>
                </div>
              ) : (
                <p className="text-gray-400">Select a PDF to display</p>
              )}
            </>
          )}
        </div>
      </div>
    </div>
  );
```

```
};

export default PDFViewer;
```

Sidebar.tsx

```
import { useState } from "react";
import { Folder, FolderOpen } from "lucide-react";

interface SidebarProps {
  onFolderSelect: (folderPath: string) => void;
  selectedFolder: string | null;
}

const Sidebar: React.FC<SidebarProps> = ({ onFolderSelect, selectedFolder }) => {
  const [folders, setFolders] = useState<string[]>([]);

  const handleOpenFolder = async () => {
    const input = document.createElement("input");
    input.type = "file";
    input.webkitdirectory = true;
    input.onchange = (event) => {
      const files = (event.target as HTMLInputElement).files;
      if (files && files.length > 0) {
        const folderPath = files[0].webkitRelativePath.split("/")[0]; // Extract folder name
        setFolders((prev) => [...prev, folderPath]);
        onFolderSelect(folderPath);
      }
    };
    input.click();
  };

  return (
    <div className="w-64 bg-gray-900 text-white p-4">
      <h2 className="text-lg font-bold mb-4 flex items-center gap-2">
        <Folder className="w-5 h-5 text-blue-400" /> My Documents
      </h2>
      <button onClick={handleOpenFolder} className="bg-blue-500 px-4 py-2 rounded
mb-4">
        <FolderOpen className="w-5 h-5 inline-block mr-2" /> Open Folder
      </button>
      <ul>
        {folders.map((folder, index) => (
          <li key={index} className={`p-2 ${folder === selectedFolder ? "bg-gray-700" : ""}`}>
            📁 {folder}
          </li>
        ))}
      </ul>
    </div>
  );
};

export default Sidebar;
```

SnippingTool.tsx

```
import React from "react";
import { Scissors } from "lucide-react";

interface SnippingToolProps {
  selectedFolder: string | null;
}

const SnippingTool: React.FC<SnippingToolProps> = ({ selectedFolder }) => {
  const handleSnip = async () => {
    if (!selectedFolder) {
      alert("⚠️ Please select a folder first.");
      return;
    }

    console.log("📂 Selected Folder:", selectedFolder);

    alert("✂️ Snipping tool activated! Select an area, and it will be auto-saved.");

    const formData = new FormData();
    formData.append("folder", selectedFolder);

    try {
      const response = await fetch("http://127.0.0.1:5000/start-snip", {
        method: "POST",
        body: formData,
      });

      const data = await response.json();
      console.log("📸 Snip Response:", data);

      if (response.ok) {
        alert(`✅ Snip saved successfully!\n📁 Location: ${data.file_path}`);
      } else {
        alert(`❌ Failed to save snip: ${data.error}`);
      }
    } catch (error) {
      console.error("❌ Error saving snip:", error);
      alert("❌ Error saving snip. Check console for details.");
    }
  };
  return (
    <button
      onClick={handleSnip}
      className="flex items-center bg-green-600 hover:bg-green-700 text-white px-4 py-2
rounded-lg"
    >
      <Scissors className="w-5 h-5 mr-2" />
      Snip & Save
    </button>
  );
};

export default SnippingTool;
```

UploadButton.tsx

```
import React, { useState } from "react";

interface UploadButtonProps {
  onUpload: (file: { filename: string; url: string }) => void;
}

const UploadButton: React.FC<UploadButtonProps> = ({ onUpload }) => {
  const [selectedFile, setSelectedFile] = useState<File | null>(null);
  const [uploadStatus, setUploadStatus] = useState<string | null>(null);

  const handleFileChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    if (e.target.files && e.target.files.length > 0) {
      const file = e.target.files[0];
      setSelectedFile(file);
      console.log("✅ Selected file:", file.name);
    }
  };

  const handleFileUpload = async () => {
    if (!selectedFile) {
      setUploadStatus("❌ No file selected.");
      console.error("❌ No file selected.");
      return;
    }

    const formData = new FormData();
    formData.append("pdf", selectedFile); // Ensure this matches backend key

    try {
      const response = await fetch("http://127.0.0.1:5000/upload-pdf", {
        method: "POST",
        body: formData,
      });
      if (!response.ok) {
        throw new Error(`HTTP error! Status: ${response.status}`);
      }

      const data = await response.json();
      console.log("✅ File uploaded successfully:", data);

      onUpload({ filename: selectedFile.name, url: data.url });

      setUploadStatus("✅ File uploaded successfully!");
    } catch (error) {
      console.error("❌ Error uploading file:", error);
      setUploadStatus("❌ Upload failed.");
    }
  };

  return (
    <div className="flex flex-col items-center space-y-4 p-4 bg-gray-900 text-white
rounded-lg shadow-lg">
      <input
        type="file"
        accept="application/pdf"
        onChange={handleFileChange}
        className="p-2 border border-gray-600 rounded-lg"
      />
      <button
        onClick={handleFileUpload}
        className="bg-green-600 hover:bg-green-700 px-4 py-2 rounded-lg text-white
font-bold transition"
      >
        Upload PDF
      </button>
      {uploadStatus && <p className="text-sm">{uploadStatus}</p>}
    </div>
  );
};

export default UploadButton;
```

UploadPDF.tsx

```
import React, { useState } from "react"; // ✅ Add useState import

type UploadPDFProps = {
  onUpload: (file: { name: string; folder: string }) => void;
};

const UploadPDF = ({ onUpload }: UploadPDFProps) => {
  const [selectedFile, setSelectedFile] = useState<File | null>(null);

  const handleFileChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    const file = event.target.files?.[0];
    if (file) {
      setSelectedFile(file);
      const folder = "uploaded-pdfs"; // Backend folder where files are stored
      onUpload({ name: file.name, folder });
    }
  };

  return (
    <div>
      <label className="block text-sm font-medium text-gray-700">Choose File:</label>
      <input type="file" accept=".pdf" onChange={handleFileChange} className="p-2 border
border-gray-300 rounded" />
      {selectedFile && <p className="text-blue-500 mt-2">Selected File:
{selectedFile.name}</p>}
    </div>
  );
};

export default UploadPDF;
```

App.tsx

```
import React, { useState, useEffect } from "react";
import Navbar from "./components/Navbar";
import Sidebar from "./components/Sidebar";
import PDFViewer from "./components/PDFViewer";
import UploadButton from "./components/UploadButton";

interface PdfFile {
  filename: string;
  url: string;
}

const App: React.FC = () => {
  const [pdfList, setPdfList] = useState<PdfFile[]>([]);
  const [folderPdfs, setFolderPdfs] = useState<PdfFile[]>([]);
  const [selectedFolder, setSelectedFolder] = useState<string | null>(null);

  useEffect(() => {
    fetch("http://127.0.0.1:5000/get-pdfs")
      .then((res) => res.json())
      .then((data) => setPdfList(data))
      .catch((error) => console.error("Error fetching PDFs:", error));
  }, []);
  const handleFolderSelect = (folderPath: string) => {
    setSelectedFolder(folderPath);

    fetch(`http://127.0.0.1:5000/get-pdfs?folder=${encodeURIComponent(folderPath)}`)
      .then((res) => res.json())
      .then((data) => setFolderPdfs(data))
      .catch((error) => console.error("Error fetching folder PDFs:", error));
  };

  const handleUpload = (newPdf: PdfFile) => {
    if (selectedFolder) {
      setFolderPdfs((prev) => [...prev, newPdf]);
    } else {
      setPdfList((prev) => [...prev, newPdf]);
    }
  };

  return (
    <div className="h-screen flex flex-col bg-gray-800 text-white">
      <Navbar />
      <div className="flex flex-1">
        <Sidebar onFolderSelect={handleFolderSelect} selectedFolder={selectedFolder} />
        <div className="flex flex-col flex-1 p-4">
          <UploadButton onUpload={handleUpload} />
          <PDFViewer pdfList={selectedFolder ? folderPdfs : pdfList}
selectedFolder={selectedFolder} />
        </div>
      </div>
    </div>
  );
};

export default App;
```

main.tsx

```
import React from "react";
import ReactDOM from "react-dom/client"; // Correct import for React 18
import App from "./App";
import "./index.css";

// Create the root element to mount the React app
const root = ReactDOM.createRoot(document.getElementById("root") as HTMLElement);

// Render the app inside the root element
root.render(
  <React.StrictMode>
```

```
    <App />
  </React.StrictMode>
);
```

app.py

```
from flask import Flask, request, jsonify, send_from_directory
from flask_cors import CORS
import os
import time
import shutil
import subprocess

app = Flask(__name__)
CORS(app)

UPLOAD_FOLDER = os.path.abspath("uploads")
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
app.config["UPLOAD_FOLDER"] = UPLOAD_FOLDER

# ✅ Upload PDFs
@app.route("/upload-pdf", methods=["POST"])
def upload_pdf():
    if "file" not in request.files:
        return jsonify({"error": "No file part"}), 400

    file = request.files["file"]
    folder = request.form.get("folder", "").strip()

    if file.filename == "":
        return jsonify({"error": "No selected file"}), 400

    folder_path = os.path.join(app.config["UPLOAD_FOLDER"], folder) if folder else
app.config["UPLOAD_FOLDER"]
    os.makedirs(folder_path, exist_ok=True)

    file_path = os.path.join(folder_path, file.filename)
    file.save(file_path)

    return jsonify({
        "message": "File uploaded successfully",
        "filename": file.filename,
        "file_url": f"http://127.0.0.1:5000/uploads/{folder}/{file.filename}" if folder else
f"http://127.0.0.1:5000/uploads/{file.filename}"
    }), 200

# ✅ Fetch PDFs
@app.route("/get-pdfs", methods=["GET"])
def get_pdfs():
    folder = request.args.get("folder", "").strip()
    folder_path = os.path.join(app.config["UPLOAD_FOLDER"], folder) if folder else
app.config["UPLOAD_FOLDER"]

    if not os.path.exists(folder_path):
        return jsonify([]), 200

    files = os.listdir(folder_path)
    pdf_files = [
        {"filename": f, "url": f"http://127.0.0.1:5000/uploads/{folder}/{f}" if folder else
f"http://127.0.0.1:5000/uploads/{f}"}
        for f in files if f.endswith(".pdf")
    ]
    return jsonify(pdf_files), 200

# ✅ Serve PDFs
@app.route("/uploads/<path:folder>/<path:filename>", methods=["GET"])
def serve_pdf(folder, filename):
    folder_path = os.path.join(app.config["UPLOAD_FOLDER"], folder)
    return send_from_directory(folder_path, filename)

# ✅ 🔥 Open Snipping Tool & Detect Correct Snip
@app.route("/start-snip", methods=["POST"])
def start_snip():
    selected_folder = request.form.get("folder", "").strip()

    if not selected_folder:
        return jsonify({"error": "No folder selected"}), 400

    folder_path = os.path.join(app.config["UPLOAD_FOLDER"], selected_folder)
    os.makedirs(folder_path, exist_ok=True)

    print(f"📂 Selected folder: {folder_path}")

    # ✅ Get current list of files before snip
    screenshots_folder = os.path.join(os.path.expanduser("~"), "Pictures", "Screenshots")
    before_files = set(os.listdir(screenshots_folder)) if os.path.exists(screenshots_folder) else
set()

    # ✅ Open Snipping Tool
    subprocess.run("explorer ms-screenclip:", shell=True)

    # ✅ Wait for the user to take a snip (max wait: 15 sec)
    timeout = 15
    start_time = time.time()

    while time.time() - start_time < timeout:
        time.sleep(2)
        after_files = set(os.listdir(screenshots_folder)) if os.path.exists(screenshots_folder) else
set()
        new_files = after_files - before_files

        if new_files:
            latest_screenshot = max(new_files, key=lambda f:
os.path.getctime(os.path.join(screenshots_folder, f)))
            found_screenshot = os.path.join(screenshots_folder, latest_screenshot)
            print(f"📸 New snip detected: {found_screenshot}")
            break
    else:
        return jsonify({"error": "No new snip detected. Please try again."}), 500

    # ✅ Move snip to selected folder
    new_path = os.path.join(folder_path, f"snip_{int(time.time())}.png")

    try:
        shutil.move(found_screenshot, new_path)
        print(f"✅ Snip saved at: {new_path}")
        return jsonify({"message": "Snip saved successfully", "file_path": new_path}), 200
    except Exception as e:
        print(f"❌ Error moving snip: {e}")
        return jsonify({"error": f"Failed to move snip: {str(e)}"}), 500

if __name__ == "__main__":
    app.run(debug=True, port=5000)
```

models.py

```
import os
import sqlite3

DB_NAME = "annotations.db"

# ✅ Create the database and annotations table if not exists
def initialize_db():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS annotations (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            pdf_name TEXT NOT NULL,
            annotation TEXT NOT NULL
        )
    """)
    conn.commit()
    conn.close()

# ✅ Save annotation to the database
```

```
def save_annotation(pdf_name, annotation):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute("INSERT INTO annotations (pdf_name, annotation) VALUES (?, ?)",
(pdf_name, annotation))
    conn.commit()
    conn.close()

# ✅ Retrieve annotations for a specific PDF
def get_annotations(pdf_name):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute("SELECT annotation FROM annotations WHERE pdf_name = ?",
(pdf_name,))
    annotations = [row[0] for row in cursor.fetchall()]
    conn.close()
    return annotations
```

routes.py

```
from flask import Flask, request, jsonify, send_from_directory
import os

app = Flask(__name__)
UPLOAD_FOLDER = './uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

@app.route('/upload-pdf', methods=['POST'])
def upload_pdf():
    if 'file' not in request.files:
        return jsonify({"error": "No file part"}), 400
    file = request.files['file']
    if file.filename == '':
        return jsonify({"error": "No selected file"}), 400
    file.save(os.path.join(UPLOAD_FOLDER, file.filename))
    return jsonify({"message": "File uploaded successfully"}), 200

@app.route('/get-pdfs', methods=['GET'])
def get_pdfs():
    files = os.listdir(UPLOAD_FOLDER)
    return jsonify(files), 200

@app.route('/pdf/<filename>', methods=['GET'])
def get_pdf(filename):
    return send_from_directory(UPLOAD_FOLDER, filename)

if __name__ == '__main__':
    app.run(debug=True)
```

database.py

```
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()
```