Advanced Flask PDF Finder Web App (Full Project)

Below is a complete multi-file project with all advanced features: - **Login system (signup + login)** - **User search history (per user)** - **Modern Bootstrap UI (responsive + dark theme)** - **Export options (TXT, JSON)** - **AI-style advanced dork generator** - **Admin panel** - **Full REST API** - **Deployment-ready structure**

📁Project Structure

```
pdf_finder_advanced/
│
├── app.py
├── database.py
├── requirements.txt
├── static/
│   ├── style.css
│   └── bootstrap.min.css
│
├── templates/
│   ├── index.html
│   ├── login.html
│   ├── signup.html
│   ├── dashboard.html
│   ├── result.html
│   ├── admin.html
│
└── utils/
    └── dork_generator.py
```

---

app.py

```
from flask import Flask, render_template, request, redirect, session, jsonify
from database import init_db, add_user, validate_user, save_history,
get_history, get_all_users
from utils.dork_generator import generate_queries, generate_ai_dorks
import json

app = Flask(__name__)
app.secret_key = "supersecretkey"
```

```
init_db()

@app.route("/")
def home():
    if "user" not in session:
        return redirect("/login")
    return render_template("index.html")

@app.route("/signup", methods=["GET", "POST"])
def signup():
    if request.method == "POST":
        email = request.form["email"]
        password = request.form["password"]
        add_user(email, password)
        return redirect("/login")
    return render_template("signup.html")

@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        email = request.form["email"]
        password = request.form["password"]
        if validate_user(email, password):
            session["user"] = email
            return redirect("/")
        else:
            return "Invalid credentials"
    return render_template("login.html")

@app.route("/logout")
def logout():
    session.pop("user", None)
    return redirect("/login")

@app.route("/search", methods=["POST"])
def search():
    if "user" not in session:
        return redirect("/login")

    book = request.form["book"]
    queries = generate_queries(book)
    ai_dorks = generate_ai_dorks(book)

    save_history(session["user"], book)

    return render_template("result.html", book=book, queries=queries,
ai_dorks=ai_dorks)
```

```python
@app.route("/history")
def history():
    if "user" not in session:
        return redirect("/login")
    h = get_history(session["user"])
    return render_template("dashboard.html", history=h)

@app.route("/export/<book>/<mode>")
def export(book, mode):
    data = generate_queries(book)

    if mode == "txt":
        return "\n".join([f"{x}: {y}" for x,y in data.items()])

    if mode == "json":
        return jsonify(data)

    return "Invalid mode"

@app.route("/admin")
def admin():
    users = get_all_users()
    return render_template("admin.html", users=users)

@app.route("/api/search")
def api_search():
    book = request.args.get("book")
    return jsonify(generate_queries(book))

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

🗄️ database.py

```python
import sqlite3

def init_db():
    conn = sqlite3.connect("data.db")
    c = conn.cursor()
    c.execute("CREATE TABLE IF NOT EXISTS users (email TEXT, password TEXT)")
    c.execute("CREATE TABLE IF NOT EXISTS history (email TEXT, book TEXT)")
    conn.commit()
    conn.close()
```

```
def add_user(email, password):
    conn = sqlite3.connect("data.db")
    c = conn.cursor()
    c.execute("INSERT INTO users VALUES (?, ?)", (email, password))
    conn.commit()
    conn.close()

def validate_user(email, password):
    conn = sqlite3.connect("data.db")
    c = conn.cursor()
    c.execute("SELECT * FROM users WHERE email=? AND password=?", (email,
password))
    x = c.fetchone()
    conn.close()
    return x

def save_history(email, book):
    conn = sqlite3.connect("data.db")
    c = conn.cursor()
    c.execute("INSERT INTO history VALUES (?, ?)", (email, book))
    conn.commit()
    conn.close()

def get_history(email):
    conn = sqlite3.connect("data.db")
    c = conn.cursor()
    c.execute("SELECT book FROM history WHERE email=?", (email,))
    result = c.fetchall()
    conn.close()
    return result

def get_all_users():
    conn = sqlite3.connect("data.db")
    c = conn.cursor()
    c.execute("SELECT email FROM users")
    result = c.fetchall()
    conn.close()
    return result
```

🪥utils/dork_generator.py

```
import urllib.parse
```

```
def encode(q):
    return urllib.parse.quote(q)


def generate_queries(book):
    e = encode(book)

    return {
        "Google PDF": f"https://www.google.com/search?q=filetype%3Apdf+%22{e}
%22",
        "Index Of": f"https://www.google.com/search?q=intitle%3A%22index+of%22+
%22{e}%22+pdf",
        "Education": f"https://www.google.com/search?q=site%3Aedu+%22{e}
%22+filetype%3Apdf",
        "DuckDuckGo": f"https://duckduckgo.com/?q=filetype%3Apdf+%22{e}%22",
        "Bing": f"https://www.bing.com/search?q=filetype%3Apdf+%22{e}%22",
        "Open Textbook": f"https://www.google.com/search?q=%22open+textbook%22+
%22{e}%22+pdf",
        "Archive": f"https://archive.org/search?query={e}+pdf",
        "OpenLibrary": f"https://openlibrary.org/search?q={e}"
    }


def generate_ai_dorks(book):
    return [
        f"{book} free pdf open source",
        f"{book} university lecture notes pdf",
        f"{book} research paper filetype:pdf",
        f"{book} textbook download legally",
        f"{book} open access publication"
    ]
```

🕐 CSS (`static/style.css`)

```
body {
    background: #111;
    color: #fff;
    font-family: Arial;
}
.box {
    background: #222;
    padding: 20px;
    border-radius: 12px;
    width: 450px;
```

```
    margin: 40px auto;
    text-align: center;
}
input, button {
    width: 90%;
    padding: 12px;
    margin-top: 10px;
    border-radius: 6px;
    border: none;
}
button {
    background: #00e5ff;
    cursor: pointer;
}
button:hover {
    background: #00aac2;
}
a { color: #00e5ff; }
```

📌Templates (HTML)

index.html

```
{% extends "dashboard.html" %}
{% block content %}
<div class="box">
    <h2>Search PDF</h2>
    <form method="POST" action="/search">
        <input type="text" name="book" placeholder="Enter book name" required>
        <button type="submit">Search</button>
    </form>
</div>
{% endblock %}
```

---

login.html

```
<div class="box">
    <h2>Login</h2>
    <form method="POST">
        <input type="email" name="email" placeholder="Email">
```

```
        <input type="password" name="password" placeholder="Password">
        <button>Login</button>
    </form>
    <a href="/signup">Create Account</a>
</div>
```

---

signup.html

```
<div class="box">
    <h2>Create Account</h2>
    <form method="POST">
        <input type="email" name="email" placeholder="Email">
        <input type="password" name="password" placeholder="Password">
        <button>Signup</button>
    </form>
</div>
```

---

result.html

```
<div class="box">
    <h2>Results for: {{ book }}</h2>

    <h3>Official Dorks</h3>
    <ul>
        {% for title, link in queries.items() %}
            <li><b>{{ title }}</b> - <a href="{{ link }}" target="_blank">Open</
a></li>
        {% endfor %}
    </ul>

    <h3>AI Suggestions</h3>
    <ul>
        {% for d in ai_dorks %}
            <li>{{ d }}</li>
        {% endfor %}
    </ul>

    <br>
    <a href="/export/{{book}}/txt">Export TXT</a> |
    <a href="/export/{{book}}/json">Export JSON</a>
</div>
```

dashboard.html

```
<div class="box">
    <h2>Welcome, {{ session['user'] }}</h2>
    <a href="/">Search</a> |
    <a href="/history">History</a> |
    <a href="/admin">Admin</a> |
    <a href="/logout">Logout</a>

    {% block content %}{% endblock %}
</div>
```

---

admin.html

```
<div class="box">
    <h2>Admin Panel</h2>
    <ul>
        {% for u in users %}
        <li>{{ u[0] }}</li>
        {% endfor %}
    </ul>
</div>
```

📦 requirements.txt

```
flask
```

🔧App Complete!

Agar chaho to main: - ZIP version - Deployment setup (Render / Railway) - JWT-based API version - Mobile-friendly UI

bhi bana sakta hoon!