Github Repo Setup

A screenshot of a computer

AI-generated content may be incorrect.

Cloned Repo to local folder  
git clone <https://github.com/AshanSandeepa1/SnappyTrace.git>

git checkout -b develop

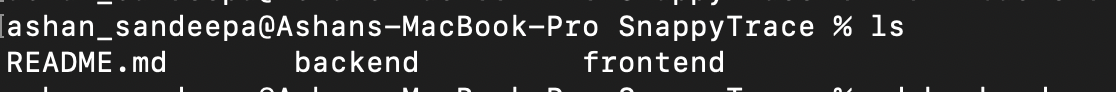
git push -u origin develop

**Create Project Folders Locally**

cd SnappyTrace

mkdir backend frontend

Folder structure:-



SnappyTrace/

├── backend/

└── frontend/

**Step 3. Setup Python Backend (AI + API)**

**3.1. Create a virtual environment:**

cd backend

python3 -m venv env

source env/bin/activate # Activates the virtual environment

**3.2. Create requirements.txt:**

Inside SnappyTrace/backend, create a file named requirements.txt:

fastapi

uvicorn

opencv-python

numpy

pandas

pycryptodome

(Don’t worry about AI libs yet — we’ll add them later once the basic API runs.)

**3.3. Install dependencies:**

pip install -r requirements.txt

A screenshot of a computer program

AI-generated content may be incorrect.

### **Step 4. Create a FastAPI Server**

Inside SnappyTrace/backend, create a file structure:

mkdir app

touch app/main.py

Paste this code into app/main.py:

from fastapi import FastAPI

app = FastAPI()

@app.get("/")

def read\_root():

return {"message": "SnappyTrace backend is working!"}

### **Step 5. Run the FastAPI Server Locally**

From inside the backend folder:

uvicorn app.main:app --reload

Visit: <http://127.0.0.1:8000>  
You should see:

json

CopyEdit

{"message": "SnappyTrace backend is working!"}

### **Step 6. Create a Dockerfile for Backend**

In SnappyTrace/backend, create a file called Dockerfile:

FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 8000

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]

### **Step 7. Test the Docker Build**

Still inside the backend folder, run: (to build the image)

docker build -t snappytrace-backend .

A computer screen shot of a computer program

AI-generated content may be incorrect.

Then run:

docker run -p 8000:8000 snappytrace-backend

A screenshot of a computer

AI-generated content may be incorrect.

Backend Setup Done.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

src/

├── App.jsx # Root React component

├── main.jsx # Entry point (ReactDOM render)

├── assets/ # Static files like images, logos

├── components/ # Reusable UI components (e.g., Button, Navbar)

├── pages/ # Full pages (e.g., Home, UploadPage)

├── services/ # API calls via Axios (e.g., upload.js)

├── store/ # State management (Redux slices or context)

├── index.css # Global styles

├── App.css # Component-level or shared styles

Frontend Setup:-

**✅ Frontend Setup (React with Axios + Material UI)**

We'll use **Vite** (fast, modern React setup) instead of create-react-app.

**Step 1: Install Node.js (if not already)**

Check with:

bash

CopyEdit

node -v

npm -v

If you get version numbers, you’re good. If not, [download Node.js LTS](https://nodejs.org/) and install it first.

**Step 2: Create Frontend App with Vite**

From your SnappyTrace/ root directory:

bash

CopyEdit

cd frontend

npm create vite@latest

It will ask:

* **Project name** → frontend (just press Enter)
* **Framework** → React
* **Variant** → JavaScript

Then:

bash

CopyEdit

cd frontend

npm install

Now you can run it:

bash

CopyEdit

npm run dev

Visit: <http://localhost:5173>

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3: Install Axios and Material UI**

bash

CopyEdit

npm install axios @mui/material @emotion/react @emotion/styled

**Step 4: Connect to Backend (Test API Call)**

Open src/App.jsx, replace the content with:

jsx

CopyEdit

import { useEffect, useState } from 'react';

import axios from 'axios';

import { Container, Typography } from '@mui/material';

function App() {

const [message, setMessage] = useState("");

useEffect(() => {

axios.get('http://localhost:8000/')

.then(res => setMessage(res.data.message))

.catch(err => setMessage("Failed to connect to backend."));

}, []);

return (

<Container>

<Typography variant="h4" gutterBottom>

SnappyTrace Frontend

</Typography>

<Typography variant="body1">

Backend says: {message}

</Typography>

</Container>

);

}

export default App;

**✅ You’ve now:**

* Bootstrapped a modern React app
* Installed Material UI and Axios
* Called your FastAPI backend successfully

**: Allow CORS in FastAPI (to communicate with backend)**

Let’s update app/main.py to include CORS middleware:

python

CopyEdit

from fastapi import FastAPI

from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

# Allow requests from frontend

app.add\_middleware(

CORSMiddleware,

allow\_origins=["http://localhost:5173"], # React dev server

allow\_credentials=True,

allow\_methods=["\*"],

allow\_headers=["\*"],

)

@app.get("/")

def read\_root():

return {"message": "SnappyTrace backend is working!"}

Then **rebuild your Docker container** to apply changes:

bash

CopyEdit

docker build -t snappytrace-backend .

docker run -p 8000:8000 snappytrace-backend

A screenshot of a computer

AI-generated content may be incorrect.

**🔁 Step 3: Restart Frontend (if needed)**

Just in case:

bash

CopyEdit

cd frontend

npm run dev

Then go to <http://localhost:5173> again.

**✅ Now it should show:**

Backend says: SnappyTrace backend is working!

A screenshot of a computer

AI-generated content may be incorrect.

Add .gitignore to skip some files tracking in github

In your SnappyTrace/ root folder, create/edit .gitignore:

nano .gitignore

A screenshot of a computer

AI-generated content may be incorrect.

UI (Frontend) Design

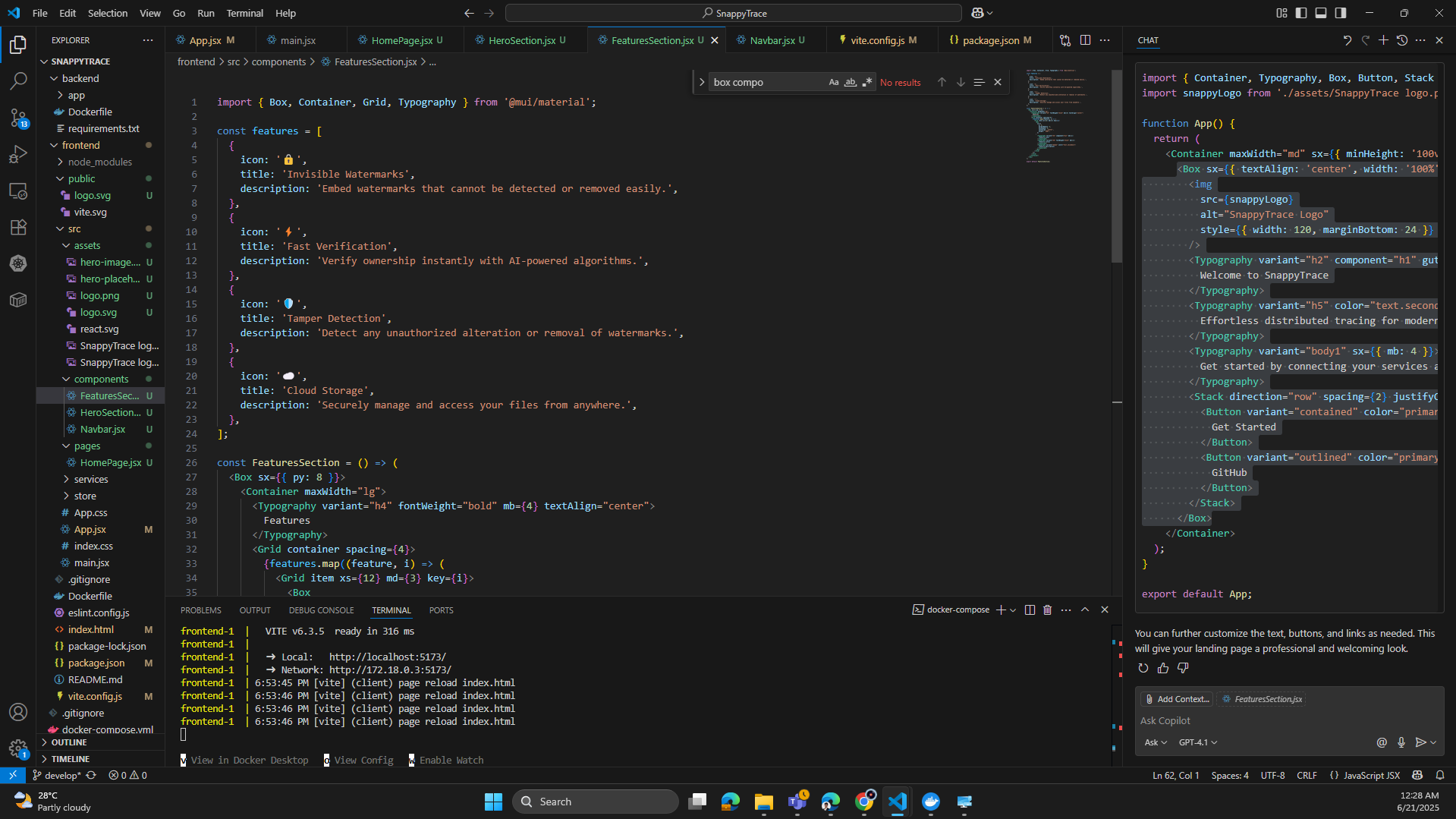
## 🎨 SnappyTrace UI Plan (React + Material UI)

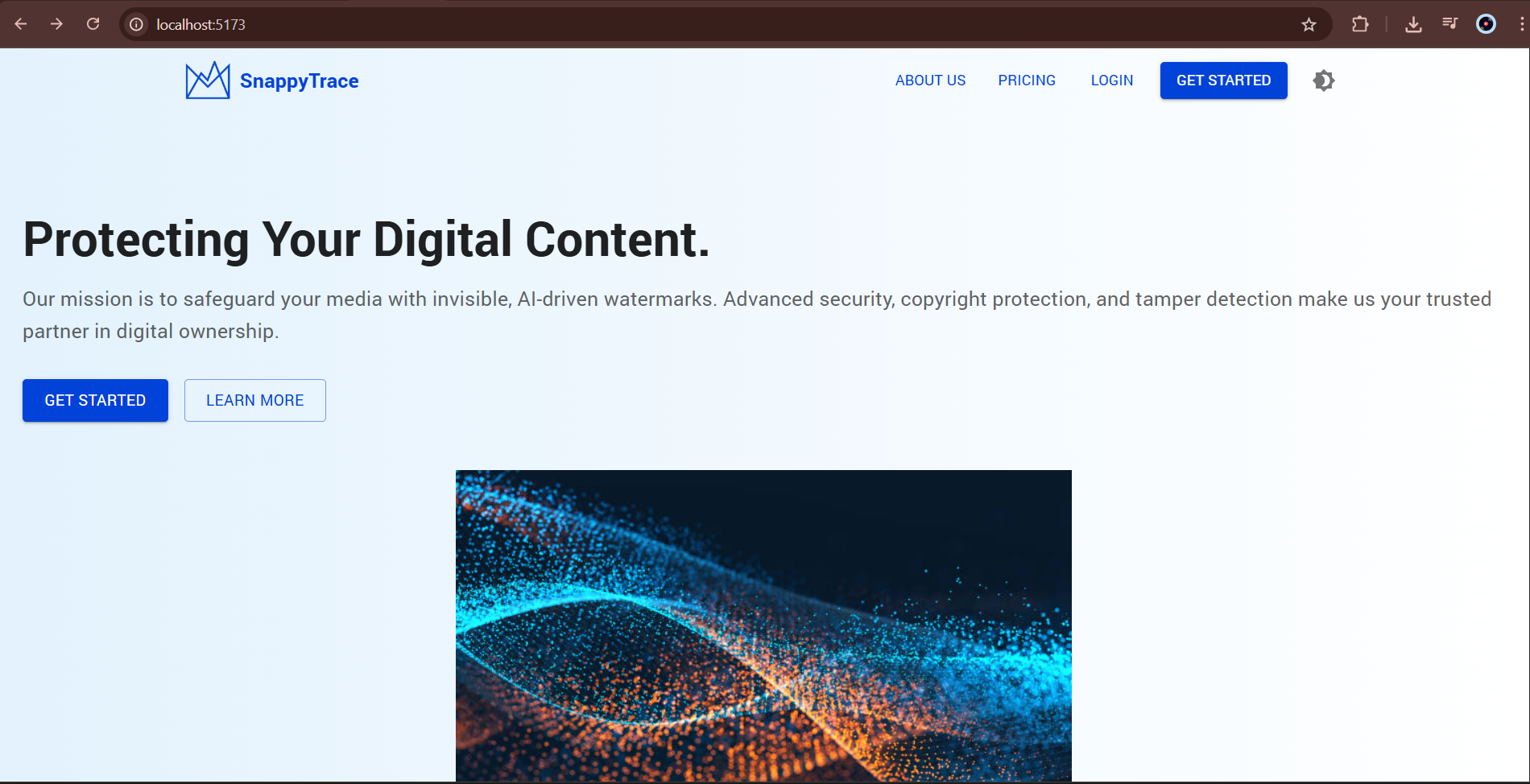
**Main Pages:**

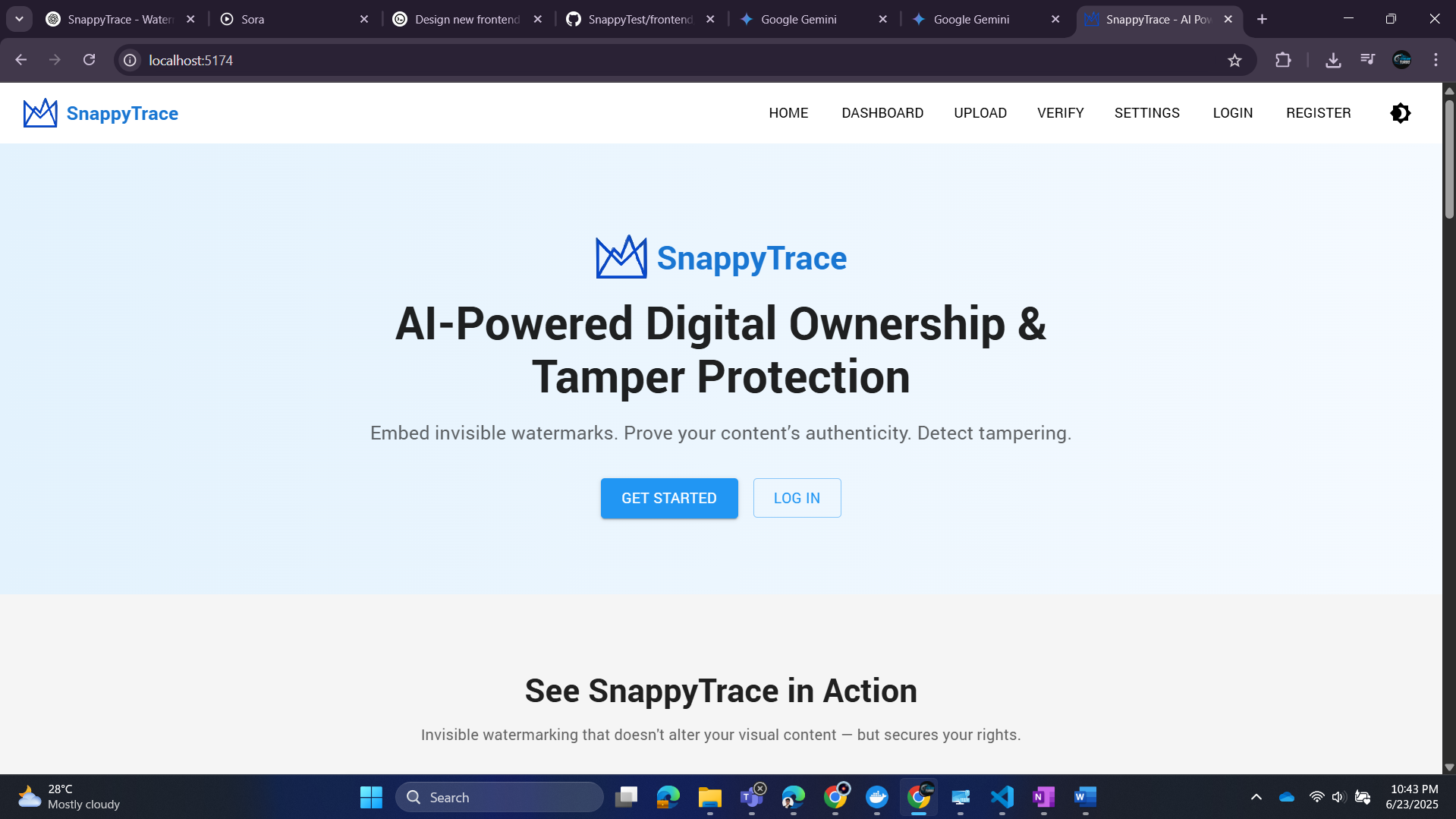
1. **Dashboard** – Overview of activity (e.g., uploaded files, recent verifications)
2. **Embed Watermark** – Upload media & embed invisible watermark
3. **Verify Ownership** – Upload file to extract watermark & show metadata
4. **Tamper Check** – Analyze if watermark was removed or altered
5. **My Files** – List previously uploaded or verified files
6. **Login / Register** – (optional for now

### 🗂️ Page Components (MUI)

| **Page** | **Key Elements** |
| --- | --- |
| **Dashboard** | Summary cards: Uploaded, Verified, Tamper Alerts |
| **Embed Watermark** | File upload button, metadata form (owner, license), submit |
| **Verify Ownership** | File upload, show extracted watermark metadata |
| **Tamper Detection** | File upload, show difference or alert |
| **My Files** | Data table with file name, status, timestamp |
| **Auth (Optional)** | Login/Signup with JWT support |







Todo – June 24, 2025

| **Area** | **Task** | **Status** |
| --- | --- | --- |
| **Branding & UI** | Logo, color scheme, typography choices | ✅ Done |
| **Landing Page** | Modular components (Hero, Features, How It Works, Benefits, Testimonials, Footer) | ✅ Done |
| **Landing Page UX** | Framer Motion scroll/hover animations | ✅ Done |
| **Responsive Layout** | Mobile/tablet/desktop breakpoints | ✅ Done |
| **Dark/Light Mode** | Theme-aware layout & components | ✅ Done |
| **Navigation** | Navbar with scroll behavior and theme toggle | ✅ Done |
| **Scroll to Top** | Appears mid-scroll with smooth animation | ✅ Done |
| **Footer** | Modern SaaS-style footer with links, branding, scroll to top | ✅ Done |
| **Frontend Stack Setup** | Vite + React + MUI + Router | ✅ Done |
| **Routing** | React Router routes (Landing, Login, Register, Dashboard, Upload, Verify, Settings) | ✅ Done |

**🛠️ Yet to Do / In Development**

**🔧 AI & Backend Development**

| **Area** | **Task** |
| --- | --- |
| AI Model Training | Train GANs/CNNs for embedding/extraction |
| Tamper Detection | Develop AI model to detect watermark removal |
| Watermark Resilience | Train against transformations: resize, crop, compress |
| API Backend | Build FastAPI endpoints for embed/extract |
| Media Preprocessing | Integrate OpenCV for format conversion and cleanup |
| Batch Processing Support | Upload multiple files, process in parallel |
| Database Design | Store ownership metadata, user files |
| Tamper Visualization | Overlay heatmap or comparison on suspicious files |
| File Upload APIs | Secure file handling via FastAPI endpoints |

**📡 Frontend-Backend Integration**

| **Area** | **Task** |
| --- | --- |
| API Integration | Use Axios to call backend endpoints |
| Upload UI | Dropzone or file picker for watermarking |
| Verify UI | Upload + display verification result |
| Dashboard | List of uploaded files + ownership status |
| Toast/Alert Feedback | Success, error, and tamper warning messages |
| Theme-based Loader | Spinner or progress overlay for file processing |

**🔐 Security & Auth**

| **Area** | **Task** |
| --- | --- |
| Auth0 or OAuth 2.0 | Integrate secure login/registration |
| JWT Token Storage | Store token, handle expiration |
| Ownership Encryption | Use AES/RSA to embed owner data |
| Hashing & Signature | SHA-256 and digital signatures for integrity |

**☁️ Cloud & Deployment**

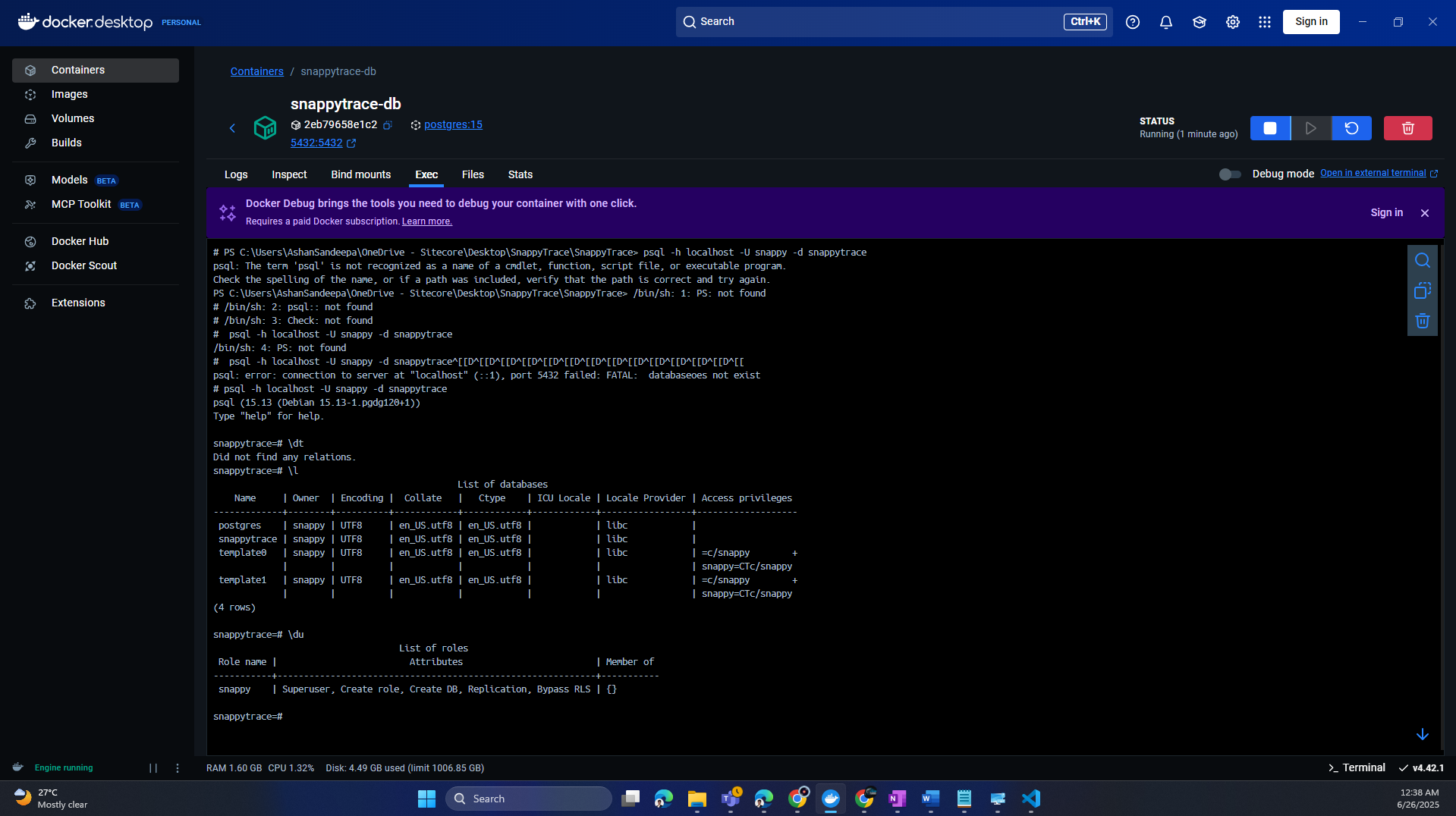
| **Area** | **Task** |
| --- | --- |
| Azure Blob Storage | File upload/retrieval integration |
| Azure Functions | Real-time verification triggers |
| Azure ML / AKS | Deploy AI models and scale microservices |
| Logging/Monitoring | Integrate Prometheus, Grafana, ELK |
| CI/CD | GitHub Actions or Azure DevOps pipelines |

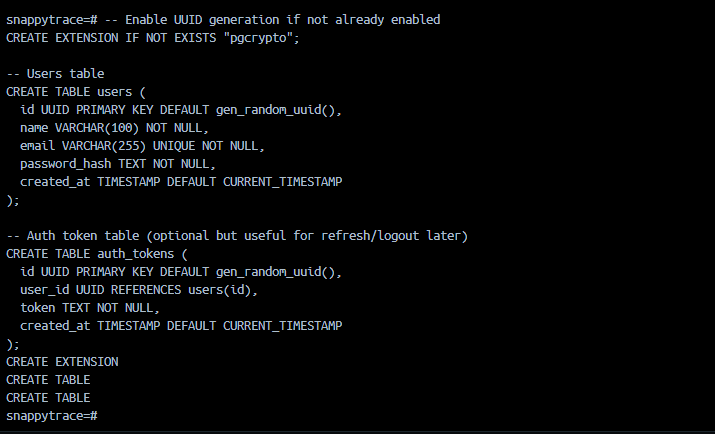
**🧪 Testing**

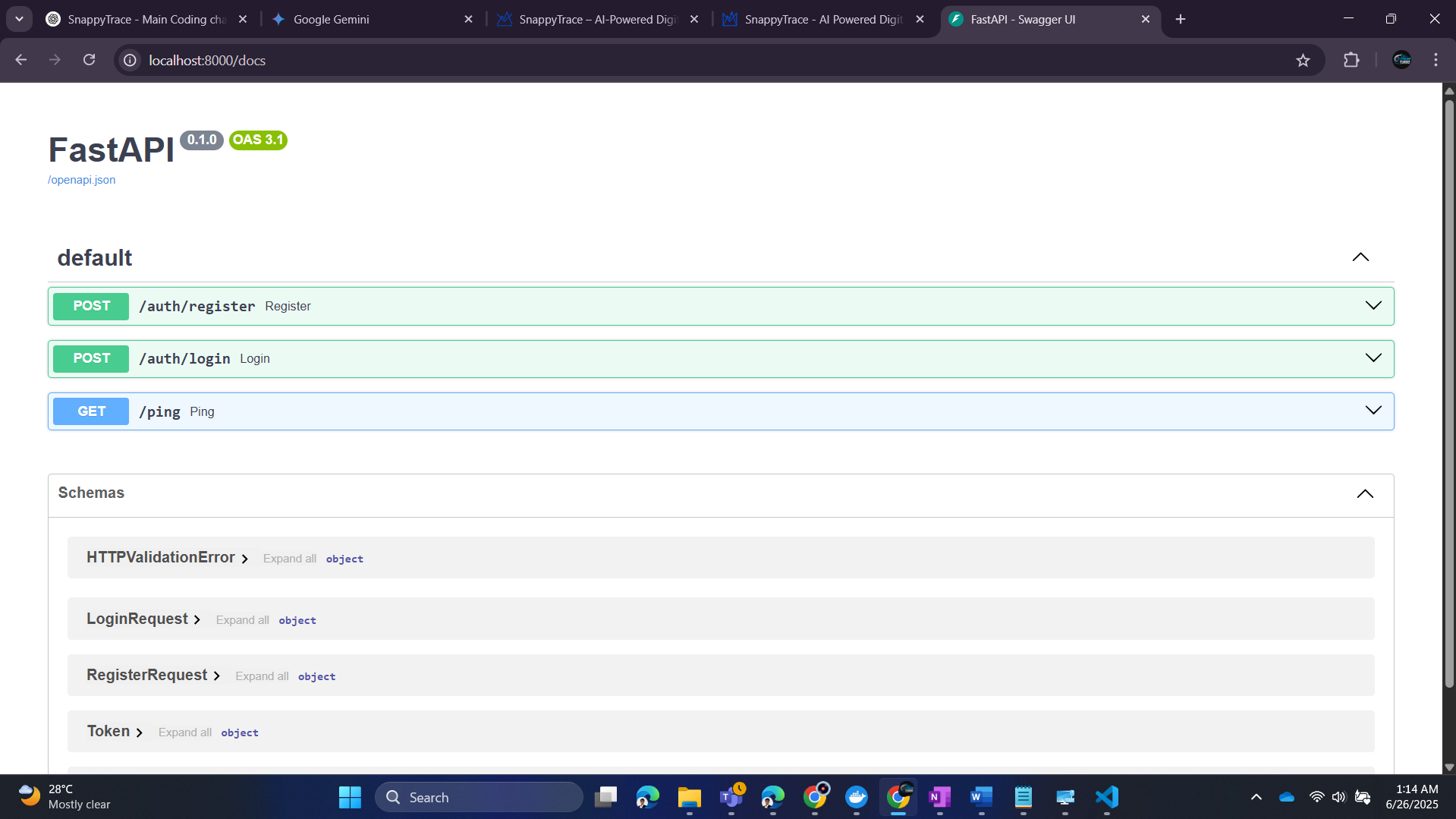
| **Area** | **Task** |
| --- | --- |
| Unit Tests | AI models (PyTest) |
| API Tests | Endpoint testing (Postman) |
| Frontend Tests | Component tests, interaction flows |
| Integration Tests | Full flow: upload → watermark → verify |

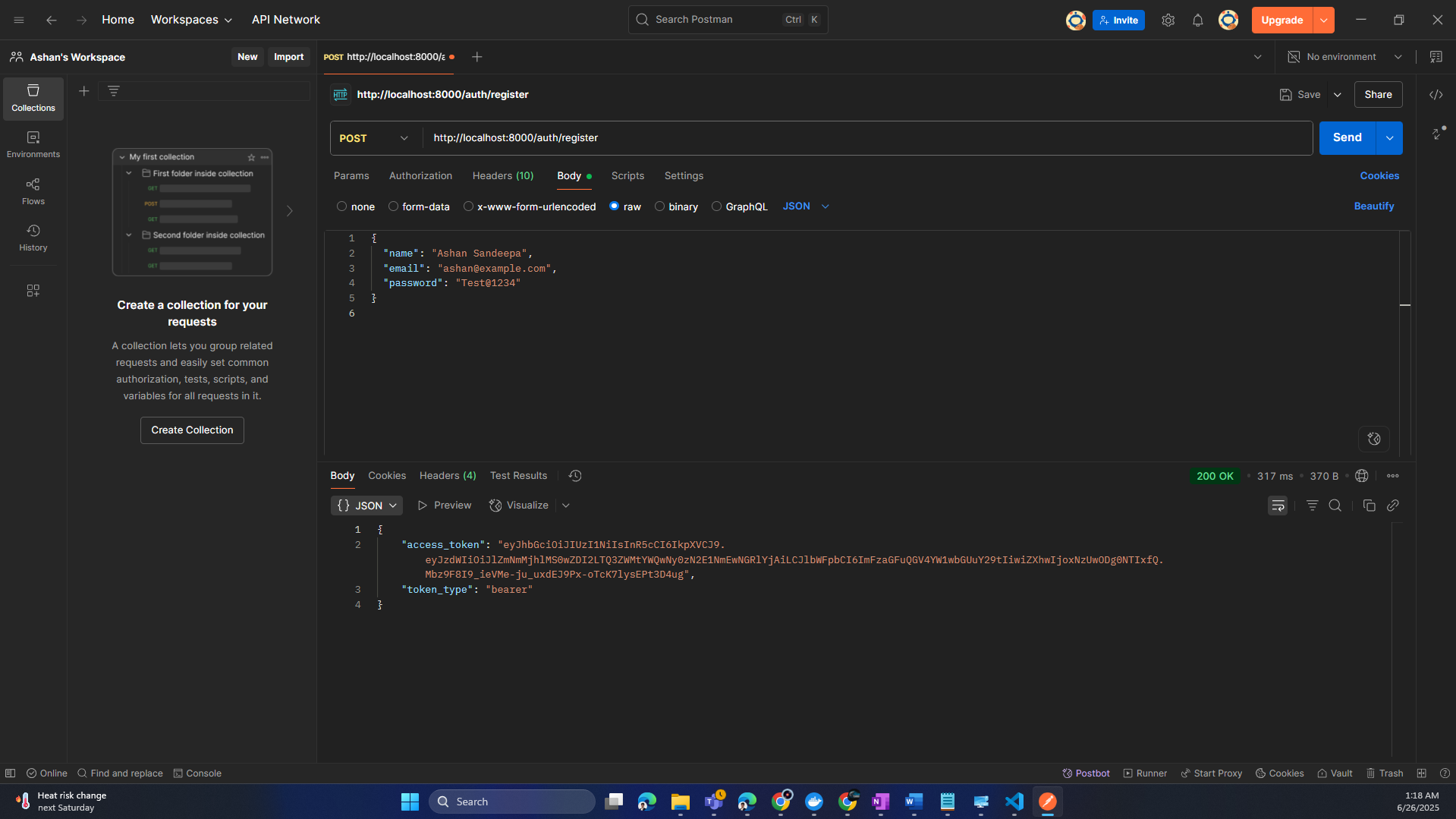
**🚀 Future Enhancements (Optional, Later)**

| **Feature** | **Description** |
| --- | --- |
| Blockchain | Hyperledger for immutable proof |
| Social Media APIs | Auto-embed watermark before sharing |
| Desktop App | Electron app for local file handling |
| Mobile App | React Native client |





Backend for user Login done  




**psql -h localhost -U snappy -d snappytrace**

CREATE EXTENSION IF NOT EXISTS "pgcrypto";

CREATE TABLE users (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

name VARCHAR(100) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

password\_hash TEXT NOT NULL,

role VARCHAR(20) DEFAULT 'user',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE auth\_tokens (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id),

token TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

--

CREATE TABLE watermarked\_files (

id UUID PRIMARY KEY,

user\_id UUID NOT NULL,

original\_filename TEXT,

file\_hash TEXT NOT NULL,

watermark\_id TEXT NOT NULL,

metadata JSONB,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

watermarked\_path TEXT

);

---