```
from flask import Flask, jsonify, request
import random

app = Flask(_name_)

# Simulated data for automation metrics
metrics = {
    "total_deployments": 10000,
    "successful_deployments": 9500,
    "failed_deployments": 500,
    "average_deployment_time": "4.5 min",
    "error_rate": "5%"
}

# Generate a large set of simulated deployment logs
deployment_logs = [
    {"id": i, "status": random.choice(["Success", "Failed"]), "time": f"{random.randint(1, 10)} min" if random.choice([True, False]) else "--"}
    for i in range(1, 10001)
]

@app.route('/')
def home():
    return "<h1>DevOps Automation Tracker</h1><p>Use /api/metrics or /api/logs to fetch data.</p>"

@app.route('/api/metrics')
def get_metrics():
    return jsonify(metrics)

@app.route('/api/logs')
def get_logs():
```

```
    print("/api/logs route accessed")  # Debug print
    page = int(request.args.get('page', 1))
    per_page = int(request.args.get('per_page', 100))
    start = (page - 1) * per_page
    end = start + per_page
    logs = deployment_logs[start:end]
    return jsonify(logs)

@app.route('/api/logs/search')
def search_logs():
    print("/api/logs/search route accessed")  # Debug print
    status = request.args.get('status')
    if status not in ["Success", "Failed"]:
        return jsonify({"error": "Invalid status"}), 400
    filtered_logs = [log for log in deployment_logs if log["status"] == status]
    return jsonify(filtered_logs[:100])  # Limiting results for performance

@app.route('/api/logs/<int:log_id>')
def get_log_by_id(log_id):
    print(f"/api/logs/{log_id} route accessed")  # Debug print
    log = next((log for log in deployment_logs if log["id"] == log_id), None)
    if log is None:
        return jsonify({"error": "Log not found"}), 404
    return jsonify(log)

if _name_ == '_main_':
    app.run(host='127.0.0.1', port=5000, debug=True)
```