AQVH

Problem statement:
Develop a tool/app that accepts a multi-qubit quantum circuit, isolates single-qubit reduced density matrices using partial tracing, and visualizes each qubit's mixed state on the Bloch sphere.

1. Objective Recap

Develop an app that:

- Accepts a multi-qubit quantum circuit.
- Simulates the circuit (statevector/density matrix).
- Performs partial trace to isolate single-qubit reduced density matrices.
- Visualizes each qubit's (potentially mixed) state on its own Bloch sphere.
- (Optionally) Adds creative, educational enhancements such as interactive UI, animations, or comic/story overlays.

---

2. Tools & Frameworks Required

Layer	**Tool/Library**	**Purpose/Justification**
Quantum Core	Qiskit	Circuit creation, simulation, partial trace, state visualization
Visualization	matplotlib	Plotting Bloch spheres (already integrates with Qiskit viz tools)

App Interface	Streamlit or Jupyter	Quick, interactive web app or notebook for input/UI
UI/Animation	SVG.js, GSAP (optional)	Comics panels, transitions, animations (web add-ons for creative interactivity)
Video Export	matplotlib.animation/Manim	Export state evolution videos (optional, for storytelling/deep insight)
Backend/ CLI	Python	Main computation language

---

3. Repository to Enhance

Recommended Base: `Qiskit` (IBM)

- Very active, robust, and extensible.
- Official repo: Qiskit GitHub
- Alternatively, start from a Streamlit or Jupyter Qiskit tutorial and create your own public repo for greater control.

You will:

- Fork/clone Qiskit for reference and enhancement.
- Or, more commonly, create your own wrapper app (repo) that imports Qiskit and adds partial tracing + enhanced visualization/UX.

---

4. Features to Implement

MVP (Minimum Viable Product)

- Upload/enter multi-qubit quantum circuits (Python, QASM, or visual builder).

- Simulate to get global state.
- Compute single-qubit reduced density matrices for all qubits (partial trace).
- Plot each qubit's mixed state on separate Bloch spheres.
- Interface (Jupyter/Streamlit) for seamless workflow.

Extensions (for strong outcomes/creativity)

- Animate state evolution or partial tracing process.
- Interactivity: sliders/controls for real-time circuit/gate tweaks.
- Comics/explainers: educational overlays or story panels.
- Export plots or videos for documentation/teaching.

---

5. Sample Workflow: End-to-End

1. User Inputs Circuit
    - Enter QASM/Python circuit or use visual builder.
2. Simulate Circuit
    - Use Qiskit Aer simulator to get statevector or density matrix.
3. Compute Reduced DMs
    - For each qubit, apply partial trace (Qiskit's `partial_trace`).
4. Visualize
    - For each reduced DM, plot Bloch sphere using `plot_bloch_multivector` or custom plot.
    - Display in grid UI; optionally animate or show purity/entropy.
5. Enhanced Output (Optional)
    - Animate changes if user tweaks input (real-time update).
    - Overlay comic panel explaining what's happening at each step.
    - Export result as image/video/interactive demo.

Example UI Workflow (Streamlit):

- User uploads/writes circuit → Clicks "Simulate"
  → Backend simulates and traces → Plots appear live
  → (Optional) Comics/explainers pop up alongside
  → Export/download results.

---

6. Main Code Skeleton (Python/Qiskit/Streamlit)

`python`

```
import streamlit as st
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit.quantum_info import DensityMatrix, partial_trace
from qiskit.visualization import plot_bloch_multivector

# 1. Input: Get circuit from user
circuit_code = st.text_area("Enter Quantum Circuit in QASM or
Python code")
# 2. Parse, simulate, get Density Matrix
qc = QuantumCircuit.from_qasm_str(circuit_code)  # add try/except
for safety
sim = AerSimulator()
result = sim.run(transpile(qc, sim)).result()
dm_full = DensityMatrix(result.get_statevector(qc))
# 3. Partial trace and visualize
figs = []
for i in range(qc.num_qubits):
    traced_dm = partial_trace(dm_full, [j for j in
range(qc.num_qubits) if j != i])
    fig = plot_bloch_multivector(traced_dm)
    figs.append(fig)
    st.pyplot(fig)
# 4. (Optional) Display comics/videos/animations
```

---

7. Suggested Folder/Repo Structure

```
text

quantum-bloch-viz/
├── app.py            # main Streamlit or Flask app
├── circuits/         # sample circuits and algorithms
├── comics/           # comic assets/SVGs/story overlays
(optional)
```

```
├── static/           # images, user assets, videos (optional)
├── README.md         # project overview, how to run
├── requirements.txt  # dependencies (Qiskit, Streamlit, etc.)
```

---

8. How to Start and Iterate

- Fork Qiskit (or just import it), then build your own app in a new repo.
- Work locally, run with `streamlit run app.py` or Jupyter for iterating quickly.
- Begin with the basic simulation and visualization before layering on creative features.
- Optionally, share on GitHub/Vercel/Streamlit Cloud for feedback and collaboration—showcase in your portfolio or resume.