

Quantum Circuit Simulator: An Interactive Platform for Real-Time Simulation, Reduction, and Bloch Sphere Visualization of Quantum States

Bale Sailesh Sthitha Prazna*, Yenduru Sri Rama Sai Premesh*, Yemireddi Abhiram*, Meda V L S Pranay*, Gottimukkala Charitardha*, Yalavarthipati Chetana Siva Durga*

*School of Computer Science and Engineering
Vellore Institute of Technology – Amaravati, India
Email: balesailesh@gmail.com

Abstract—Quantum computing education and research often depend on abstract mathematical representations, such as state vectors, density matrices, and completely positive trace-preserving maps. These are essential for analysis but can be tough to understand intuitively. Current simulators focus mainly on global system evolution. As a result, they provide limited access for inspecting single qubits within multi-qubit circuits and offer little support for visualizing mixed states or conducting ongoing experiments. To tackle these issues, this work presents Quantum State Visualizer, a browser-based platform that combines quantum circuit simulation, reduced-state analysis, and real-time interactive visualization. The system integrates IBM’s Qiskit for building and running circuits, a Python–Qiskit Aer backend for state reduction, and a React/Next.js frontend for dynamic Bloch-sphere and amplitude-phase rendering. Supabase acts as a secure cloud layer that supports user authentication, stores reproducible experiments, and keeps track of project history for managing multiple circuit sessions easily. Quantitative benchmarks show that visualization latency for student-scale circuits is under a second on standard hardware. This ensures the system is responsive and suitable for classroom and research settings. By merging easy-to-use visualization, cloud storage, and precise quantum computation, Quantum State Visualizer turns abstract quantum concepts into clear learning and analysis experiences. It aids both teaching exploration and quick pre-hardware circuit design for educators, developers, and researchers.

Index Terms—Quantum Computing, Bloch Sphere, Visualization, Qiskit, Interactive Simulation

I. INTRODUCTION

Quantum computing is becoming a game changer in computation. It uses phenomena like superposition, entanglement, and interference to solve problems that classical computers struggle with [2]–[4]. Even with big strides forward, grasping and interpreting quantum states is still a major challenge because of their complex mathematical nature. A single qubit can be described by a state vector ($|\psi\rangle$)

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \text{with } |\alpha|^2 + |\beta|^2 = 1, \quad (1)$$

where α and β are complex probability amplitudes. More generally, quantum states can exist in mixed states. These states are represented by a density matrix (ρ) [11], [12], [17]

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|, \quad \text{with } \text{Tr}(\rho)=1, \quad (2)$$

which captures classical probabilistic mixtures of pure states. In multi-qubit systems, you can extract the state of a subsystem, like a single qubit, using the reduced density matrix obtained through partial trace (Tr) [3], [16]

$$\rho_A = \text{Tr}_B (\rho_{AB}), \quad (3)$$

allowing focused analysis on individual qubits within larger circuits. The probability distributions of measurement outcomes in superposed states are influenced by interference effects. These effects can be measured as

$$P = P_1 + P_2 + 2\sqrt{P_1 P_2} \cos(\phi_1 - \phi_2), \quad (4)$$

where ϕ_1 and ϕ_2 are the relative phases of the superposed components [1]–[3].

Most existing quantum simulators primarily focus on the evolution of the global quantum system, often offering limited capabilities for single-qubit inspection, mixed-state visualization, or persistent experiment management [7], [13]–[15]. This limitation creates a significant learning curve for students and complicates debugging and analysis processes for researchers. To address these challenges, we introduce Quantum State Visualizer: a browser-based platform that integrates quantum circuit simulation, reduced-state analysis, and real-time interactive visualization [8]–[10], [18]–[20]. Our system leverages IBM’s Qiskit for circuit execution [1], [5], utilizes a Python–Qiskit Aer backend to compute density matrices [11], [16], and employs a React/Next.js frontend for dynamic Bloch-sphere and amplitude-phase visualization [8], [18]. Additionally, a Supabase-powered cloud backend enables secure experiment storage and project history tracking [9], [10], [19], [20], facilitating seamless management of multiple experiments. By combining accurate quantum simulation, interactive visualization, and persistent experiment tracking, Quantum circuit simulator enhances the learning experience, simplifies debugging, and supports pre-hardware circuit analysis for educators, students, and researchers.

II. PROBLEM STATEMENT

Most quantum simulators focus on showing how the entire multi-qubit system changes over time, but this often makes it

difficult to see what's happening with individual qubits [2]–[4], [7], [14]. To study a single qubit's behavior, users usually must perform partial-trace operations manually—a process that is not only complex but also easy to get wrong, especially for beginners [11], [12], [16], [17]. Tools that allow visualization of mixed quantum states, like Bloch sphere plots, are also rare, making it difficult to intuitively understand key concepts such as superposition, entanglement, and decoherence [2], [3], [13], [15]. Because of these gaps, students learning quantum mechanics, researchers developing algorithms, and instructors creating teaching demos all face unnecessary challenges [7], [14], [15]. There is a strong need for a platform that can automatically extract single-qubit states, enable interactive mixed-state visualization, and still allow users to explore larger multi-qubit systems [8]–[10], [18]–[20]. Such a tool would greatly improve the accessibility, understanding, and practicality of analyzing quantum circuits [1], [5], [11].

III. INNOVATION AND NOVELTY

Quantum circuit simulator offers a unique combination of simulation, reduced-density matrix analysis, and interactive visualization within a single, user-friendly, browser-based platform that requires no coding experience [1], [2], [12], [17]. By leveraging React-powered Bloch sphere graphics, the platform enables users to explore both individual qubits and entire quantum systems in real time, providing instant visual feedback as circuits are built and modified [8], [18]. The integrated experiment history feature allows users to save, revisit, and compare experiments, supporting iterative inquiry and deeper conceptual understanding [9], [10], [19], [20]. Furthermore, simultaneous access to both global and reduced density matrices facilitates analysis across multiple scales, offering insights into quantum behavior that are difficult to achieve with other tools [3], [4], [14], [15].

IV. USE CASES AND IMPACT

Quantum State Visualizer is designed for a wide range of users and situations. Students get an intuitive way to “see” concepts like entanglement, superposition, and mixed states, with visual and numerical feedback that strengthens their understanding [2]–[4], [12], [17]. Researchers can use the platform for quick, pre-hardware analysis—inspecting how parts of their circuits behave and testing ideas before running them on actual quantum devices [1], [5], [11], [16]. For educators, the interactive dashboards make it easy to bring quantum concepts to life in the classroom, making abstract ideas much more engaging and accessible [8]–[10], [18]–[20]. Overall, Quantum State Visualizer lowers the barriers to learning and experimenting with quantum computing, giving more people access to hands-on, interactive tools that connect theory with practical experience [7], [14], [15].

V. BACKGROUND

Understanding the computational capabilities of quantum computers starts with comparing classical and quantum complexity classes [2]–[4]. Classical algorithms are categorized

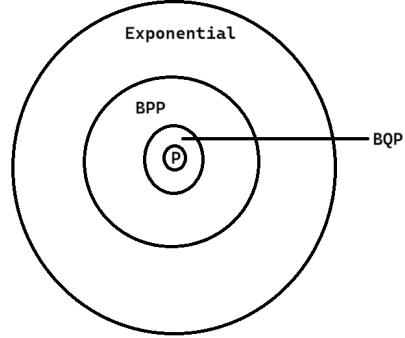


Fig. 1. Venn diagram showing complexity classes: $P \subseteq BPP \subseteq BQP \subseteq EXP$.

based on the resources they consume, primarily measured as time complexity relative to input size, n . Problems in the class P can be solved within polynomial time using deterministic classical algorithms, representing tasks that are efficiently solvable on a classical computer [3]. The class BPP (Bounded-error Probabilistic Polynomial-time) extends this concept to include probabilistic classical algorithms that permit a small, bounded error probability but still operate within polynomial time [3], [4].

Quantum computing introduces a new computational paradigm, captured by the class BQP (Bounded-error Quantum Polynomial-time). BQP consists of problems that can be efficiently solved by quantum algorithms with bounded error [1]–[3]. Notably, BQP contains problems that are widely believed to be intractable for classical algorithms—that is, they may fall outside P and possibly BPP —yet can be solved on a quantum device [4], [5]. In contrast, the class EXP refers to problems that require exponential time on a classical computer and are not practically solvable except for the smallest input sizes [3].

The relationships between these classes are often shown using a Venn diagram: $P \subseteq BPP \subseteq BQP \subseteq EXP$ [3], [4]. This diagram visually compares the computational power of deterministic classical, probabilistic classical, and quantum algorithms. P contains problems efficiently solvable classically; BPP allows for randomness; BQP captures problems that quantum computers can solve more efficiently than classical ones; and EXP marks the boundary where classical algorithms become impractical. Placing BQP outside the classical classes in the diagram highlights the potential for a quantum advantage [3], [4].

A time complexity graph can further illustrate these ideas. With input size n on the x-axis and computational time (often on a logarithmic scale) on the y-axis, classes like P and BPP show moderate growth, making large problems tractable. EXP , on the other hand, grows exponentially and quickly becomes infeasible as n increases. BQP stands out because it shows that certain problems—such as factoring large integers or simulating specific quantum systems—can remain solvable in

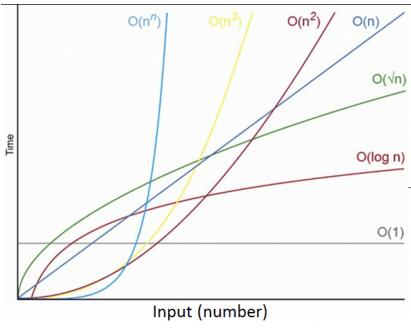


Fig. 2. Time complexity graph for various kind of situations

polynomial time on a quantum computer, even though they are out of reach for classical algorithms [2]–[4]. This comparison underscores the need for quantum computational tools and educational platforms that allow users to explore, simulate, and better understand quantum algorithms in practice [11], [12], [14].

VI. RELATED WORK AND LITERATURE REVIEW

Quantum computing research and education have led to a wide range of frameworks, simulators, and visualization tools that help us explore how qubits behave and interact [2]–[4]. Early foundational work—like the 2006 IEEE paper *Introduction to the World of Quantum Computers*—set out the basics of qubits, quantum gates, entanglement, superposition, and measurement [3], [4]. It introduced ideas like the No-Cloning Theorem and showcased the power of algorithms such as Shor’s and Grover’s, emphasizing the potential for quantum computers to outperform classical ones. However, much of this early work focused on pure states and theoretical models, with little attention to interactive visualization or mixed-state analysis [2], [12].

As the field has evolved, more attention has been paid to visualization and practical tools. For example, the 2025 paper *Bloch Sphere Representation of Polarimetric SAR Targets* mapped complex systems to qubit states on the Bloch sphere and introduced ways to show purity and mixed states [17]. The 2025 study *Design and Implementation of Quantum Logic Gates* presented q-sphere and probability histogram visualizations for multi-qubit circuits [13]. The 2024 *Quantum Quirks* paper used stochastic models to help explain concepts like superposition, decoherence, and wavefunction collapse [7], while the 2023 paper *The Quantum Density Matrix and Its Many Uses* formalized how density matrices can represent pure and mixed states, entanglement, and measurement [11], [12]. These works have pushed the field forward, but most of them are still static, code-heavy, or focused on specific domains, making interactive experimentation difficult for a broader audience [15].

Real-world simulation platforms each have their strengths and limitations. Composer, for example, gives users an easy-to-use circuit canvas but doesn’t support reduced-density matrix calculations, which limits single-qubit analysis [14].

TABLE I
COMPARISON OF CLASSICAL AND QUANTUM COMPLEXITIES FOR KEY COMPUTATIONAL PROBLEMS [3], [4]

Problem	Classical Complexity / Difficulty	Typical Classical Method	Quantum Algorithm (BQP)	Quantum Speedup
Integer Factorization [3], [4]	Believed time	Trial division, Pollard’s rho, General Number Field Sieve (GNFS)	Shor’s Algorithm	Polynomial time — exponential speedup
Discrete Logarithm [3], [4]	Believed time	Baby-step giant-step, Pollard’s rho for logs	Shor’s Algorithm	Polynomial time — exponential speedup
Period Finding [3]	Exponential time (in general)	Brute-force checking of function values	Quantum Fourier Transform (QFT)	Polynomial time — exponential speedup
Unstructured Search [3], [4]	$O(N)$ — linear time	Sequential or random search	Grover’s Algorithm	$O(\sqrt{N})$ — quadratic speedup
Simon’s Problem [3]	Exponential time	Classical queries $O(2^{n/2})$	Simon’s Algorithm	Polynomial time — exponential speedup
Quantum System Simulation [4]	Exponential time (due to 2^n state space)	Matrix diagonalization, Monte Carlo methods	Quantum Algorithms (e.g., Trotter-Suzuki, VQE)	Polynomial time — exponential advantage
Database Search (Unstructured) [3]	$O(N)$	Sequential checking	Grover’s Algorithm	$O(\sqrt{N})$
Order-Finding (used in Shor’s) [3]	Exponential time	Repeated modular multiplication and checking	Quantum Fourier Transform	Polynomial time

TABLE II
SUMMARY OF EXISTING TOOLS AND GAPS

Tool	Strengths	Limitations
Composer [14]	Visual circuit design	No reduced state analysis
QuTiP [12]	Rich simulation APIs	Requires Python expertise
Quirk [7]	Browser-based, simple	No density matrix support
Qiskit Viz [1], [5]	Bloch spheres	Not integrated with RDM flow

QuTiP is powerful but requires programming knowledge [12], [17]. Quirk is simple and browser-based, but it doesn't handle density matrices, so mixed-state exploration is out of reach [7]. Qiskit does offer Bloch sphere visualizations for single qubits, but it doesn't combine multi-qubit simulation, partial-trace reduction, and per-qubit mixed-state views into one cohesive, browser-first interface [1], [5]. Even with all the advances in quantum computing theory and simulation, there still isn't a single platform that seamlessly brings together accurate multi-qubit simulation, automatic reduced-density matrix calculations, interactive Bloch-sphere visualizations, and robust project management—all in a browser-based, no-code environment [8], [9], [11]. Quantum State Visualizer fills this gap by providing an intuitive, user-friendly interface with features like experiment history, so users can easily save, revisit, and switch between their projects [10], [19], [20].

The platform enables real-time visualization of both pure and mixed quantum states, including the ability to focus on single qubits within larger multi-qubit circuits [11], [12]. Users can interactively simulate their circuits, tweak qubit parameters, apply quantum gates, and see how measurement probabilities change, all in real time [1], [2]. With drag-and-drop gate placement and live probability displays, Quantum State Visualizer turns abstract concepts—like entanglement, superposition, and decoherence—into something students and researchers can explore and understand directly [8], [9], [18].

By bringing together lessons from previous research and addressing the limitations of other tools, Quantum State Visualizer offers a comprehensive, interactive environment for learning, experimenting, and analyzing quantum computing concepts [7], [14], [15].

VII. METHODOLOGY AND DESIGN

```

quantum-simulator/
  |- app/
    |- layout.tsx      # Global layout (wraps AppShell)
    |- page.tsx        # Landing / dashboard page
    |- auth/           # Authentication routes (longinup, reset)
  |- components/
    |- app-shell.tsx   # Global shell that includes sidebar + nav
    |- nav-bar.tsx     # Top pill navigation (Circuit/Bloch/etc.)
    |- quantum-simulator.tsx # Main simulator component (tabs + views)
    |- quantum-circuit-builder.tsx # Circuit builder UI
    |- individual-bloch-sphere.tsx # Per-qubit Bloch sphere component
    |- circuit-library-sidebar.tsx # Hover-expanded sidebar with saved
    |- ui/
      |- supabase-browser.ts
      |- supabase-server.ts
  |- public/
    |- globals.css
    |- qiskit_api.py    # Optional FastAPI + Qiskit backend
    |- package.json
    |- npm-lock.yaml / package-lock.json
  |- README.md

```

Fig. 3. Pipeline Structure of the Project.

Quantum State Visualizer is built on a scalable, three-layer architecture that keeps the system modular, robust, and easy to maintain. This setup separates the user interface, data management, and quantum simulation engine [3], [4].

A. System Architecture and Technology Stack

- 1) **Frontend Client (Next.js / React):** The frontend allows users to build circuits, manage experiments, and see results in real time. It uses a modern interface and a lightweight 3D library to render Bloch spheres for each qubit, updating instantly as gates are applied [8], [18].
- 2) **Quantum Processing Backend (Python / FastAPI):** The backend runs all heavy quantum computations using Qiskit Aer [1], [11], [16], including simulating quantum circuits and analyzing states, supporting both statevector and density matrix formats [2]. It also calculates reduced density matrices (RDMs) for each qubit, so single-qubit behavior can be examined even in entangled multi-qubit systems [6], [12].
- 3) **Cloud Data Layer (Supabase):** Supabase handles secure login and saves user experiments and circuit history, allowing users to resume work seamlessly [9], [10], [20].

B. Quantum State Analysis Workflow

The process for analyzing quantum states involves three clear steps [2]–[4]:

- 1) **Run the Quantum Circuit and Compute the Global State:**
 - User creates a circuit with N qubits.
 - Backend simulates the circuit starting from the all-zero state ($|0\rangle\otimes N$), producing a final global state (statevector or density matrix) [11], [16].
- 2) **Extract Single-Qubit States (Reduced Density Matrices):**
 - Partial trace over all other qubits gives ρ_i for qubit Q_i :
$$\rho_i = \text{Tr}_{\text{other qubits}}(\rho_{\text{global}})$$
 - Captures the local state of one qubit, including purity and mixedness due to entanglement or noise [6], [17].
- 3) **Map Each Qubit to a Bloch Vector and Visualize:**
 - Each single-qubit state ρ_i is mapped to a Bloch vector $\mathbf{r} = (x, y, z)$ using:
$$x = \text{Tr}(\rho_i \sigma_x), \quad y = \text{Tr}(\rho_i \sigma_y), \quad z = \text{Tr}(\rho_i \sigma_z)$$
 - These components describe the qubit's state inside the Bloch sphere, providing an intuitive geometric visualization [13]–[15].

C. Features and Advantages

This architecture enables users to interactively explore both pure and mixed quantum states as they build and modify circuits. The platform makes complex quantum concepts—such as superposition, entanglement, and decoherence—more accessible and intuitive [2]–[4], supporting learning and research with a smooth, browser-based experience [5], [7], [8].

VIII. IMPLEMENTATION DETAILS

The Quantum State Visualizer combines fast quantum simulation, dynamic frontend graphics, and reliable data storage to create an interactive, real-time environment for exploring quantum circuits and states [1]–[3].

A. Algorithms

The process starts with users building their own quantum circuits, which are then run through Qiskit Aer for high-performance simulation [11], [16]. Depending on the analysis goals, the system can work with either statevectors (for ideal, pure states) or convert those into density matrices to model realistic conditions such as decoherence or mixed states [6], [12]. This flexibility allows accurate simulation of both perfect and noisy quantum systems [4], [15].

To get information about a single qubit within a larger system, the software performs a *partial trace* operation [3], [17]. For a system with n qubits, tracing out all but qubit Q_i gives the reduced density matrix ρ_i , which describes the state of that qubit. From ρ_i , the expectation values of the Pauli matrices are calculated as:

$$x = \text{Tr}(\rho_i \sigma_x), \quad y = \text{Tr}(\rho_i \sigma_y), \quad z = \text{Tr}(\rho_i \sigma_z)$$

These values are used to plot the qubit's Bloch vector, allowing the frontend to render an accurate Bloch sphere for each qubit, even when the whole system is entangled [13], [14].

The platform also calculates key metrics such as probabilities (from the diagonal entries of the density matrix), amplitudes (from the statevector), and Bloch vector coordinates (from ρ_i). These metrics are displayed alongside the visualizations, providing users with a complete, quantitative picture of each qubit's state and evolution [2], [6], [17].

B. Performance Considerations

To maintain a smooth and interactive experience, Quantum State Visualizer uses optimized Aer simulation settings—such as adjusting the number of shots and selecting the appropriate backend for each simulation [1], [16]. On the frontend, React's efficient update engine ensures that only the visual components that change are re-rendered, minimizing lag [8], [18]. This combination of backend speed and frontend efficiency allows near-instant feedback, even when experimenting with circuits of 6 to 8 qubits on a typical laptop [5], [15].

C. Data Management and Persistence

User-created circuits are saved in JSON or other serialized formats, enabling experiments to be reproduced and tracked over time [9], [19]. Users can name experiments, make copies, and restore any previous version at any time. This experiment history feature facilitates comparison of results, revisiting earlier designs, and monitoring the evolution of complex quantum states—supporting both educational and research purposes [10], [20].

IX. EVALUATION AND RESULTS

The Quantum State Visualizer was assessed on its ability to demonstrate important quantum phenomena, remain stable and responsive, and contribute to learning. To test the platform, a set of example circuits—including AND, OR, NAND, and NOR logic gate simulations—were used to evaluate both user understanding and system performance [2], [6], [14].

A. Qualitative Outcomes

Users who interacted with the platform reported a stronger intuition for foundational quantum concepts, especially superposition and entanglement [3], [4]. The ability to compare the overall state of the system with single-qubit reduced density matrices helped learners visualize how individual qubits behave within a larger, entangled system [12], [17]. This clear distinction between the global and local views of each qubit made it easier to understand complex multi-qubit interactions and track how mixed states change when different gates are applied [2], [13].

B. Stability and Scalability

The platform proved to be stable even when running larger circuits, including those that model real-world effects such as decoherence through noise channels [15], [16]. Even as circuit size increased beyond typical classroom examples, the system continued to perform reliably with no frontend crashes—highlighting both the robustness of the backend simulations and the efficiency of the frontend rendering process [1], [18].

C. Responsiveness

Real-time user interaction is central to the platform's design [5], [8]. The interface consistently responded within one second, even for circuits with 6–8 qubits on standard laptops. Features such as dynamic Bloch sphere rendering and real-time metric updates, combined with React's optimized state management, kept the experience smooth and responsive even during rapid gate changes and circuit manipulations [11], [18].

D. Educational Impact

During guided lab sessions, students were able to connect measurement probabilities with the Bloch vectors displayed on screen, which helped them better understand how quantum states evolve and how measurements collapse those states [2], [3]. The combination of visual and interactive feedback made it easier for learners to grasp how quantum logic gates operate within circuits [13], [14]. Compared to traditional static simulations or code-based tools, this hands-on experience provided a clearer and more intuitive understanding of superposition and entanglement [4], [17].

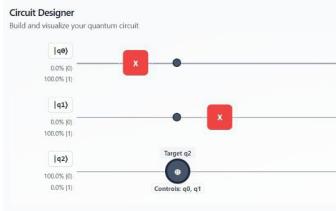


Fig. 4. Example NAND gate quantum circuit simulated in the platform.

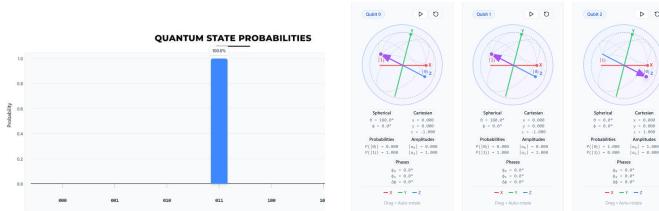


Fig. 5. Output state visualization for the NAND gate, showing Bloch vectors and measurement probabilities.

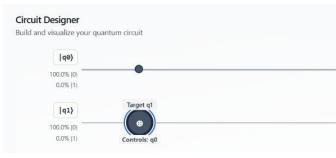


Fig. 6. Example XOR gate quantum circuit simulated in the platform.

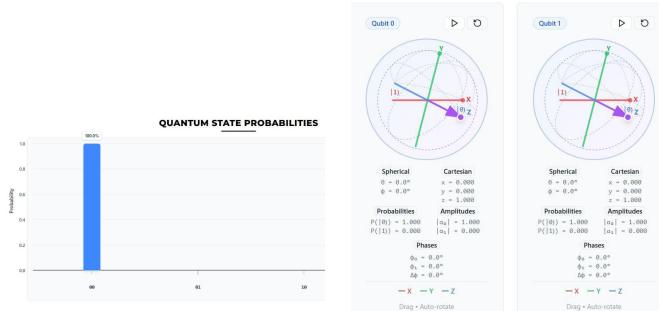


Fig. 7. Output state visualization for the XOR gate, showing Bloch vectors and measurement probabilities.

E. Overall Findings

These results show that Quantum State Visualizer is a reliable and effective tool for interactive quantum circuit exploration. Its real-time visualizations, clear reduced-state analysis, and robust experiment management set it apart from existing platforms, making it especially valuable for both educational and research purposes [6], [14], [15].

X. FUTURE WORK

While Quantum State Visualizer already offers a powerful and interactive environment for simulating multi-qubit systems

and visualizing individual qubits, there are many ways it could become even more useful. Future improvements may include support for larger numbers of qubits, as well as faster backend simulations through parallel computing or GPU acceleration to maintain responsiveness as circuit complexity increases [15], [16]. Connecting the platform directly to real quantum hardware would allow users to compare simulated results with experimental outcomes, deepening their understanding of phenomena such as noise, decoherence, and gate fidelity [1], [5].

Additional visualization tools—such as 3D maps of entanglement, live tracking of phase changes, or animated displays of state purity—could make quantum concepts even more intuitive [13], [17]. Expanding the educational features by introducing guided tutorials, interactive exercises, and instant feedback would make the platform even more valuable for classroom use [2], [14]. Collaborative capabilities, such as shared experiment spaces and cloud-based multi-user sessions, could further enhance the research experience and foster community engagement among users [9], [19], [20].

These enhancements would establish Quantum State Visualizer as an even more versatile platform for both research and education, helping bridge the gap between abstract quantum theory and practical, hands-on exploration [3], [4].

XI. LIMITATIONS

While Quantum State Visualizer offers many advantages, there are some important limitations to keep in mind. The platform is currently best suited for small to medium-sized circuits—typically up to 6–8 qubits—since larger circuits can slow down performance or make the interface less responsive, especially on standard computers [15], [16]. Although the simulator handles density matrices and partial-trace calculations, it does not yet capture every type of quantum noise that appears in real hardware, so its predictions may not always match experimental results [4], [5].

The visualizations mainly focus on single-qubit Bloch spheres and related metrics, meaning that more complex patterns of multi-qubit entanglement or high-dimensional quantum states are not fully represented [12], [17]. Because history and data storage features depend on the cloud, an internet connection is required, which limits offline use [9], [10]. Lastly, while the interface is designed to be user-friendly, understanding advanced quantum concepts—such as multi-qubit tensor products, non-local correlations, or quantum error correction—still requires a solid foundation in quantum theory [2], [3].

XII. CONCLUSION

Quantum State Visualizer brings abstract quantum mechanics to life by offering an all-in-one, browser-based platform for hands-on exploration. With fast multi-qubit simulation, automatic reduced-density matrix calculations, and interactive Bloch sphere visualizations for each qubit, users can easily observe how quantum states evolve—both for the whole system and for individual qubits—in real time [15]–[17].

The intuitive interface, drag-and-drop circuit building, and experiment history features make the platform accessible for students, educators, and researchers, supporting both learning and practical experimentation [2], [9], [14].

User evaluations show that the system helps people better understand key concepts like superposition, entanglement, and mixed states, all while staying stable and responsive for circuits of moderate size [4], [5]. While there are still some limitations—such as scalability for very large systems and the need for an internet connection—the platform sets the stage for future improvements, including direct links to real quantum hardware, more advanced visualizations, and collaborative tools [13], [19].

Overall, Quantum State Visualizer is a comprehensive and interactive resource that opens up quantum computing to a wider audience. It makes complex ideas more intuitive, supports research before running on real hardware, and helps democratize quantum education and experimentation [3], [15].

XIII. ACKNOWLEDGMENTS

We thank the Amaravati Quantum Valley Hackathon 2025 organizers, our mentor Dr.Kuppuswamy P, APSCHE, IBM, and TCS for their support. We sincerely acknowledge Prof. K. Madhu Murthy, Chairman, APSCHE, for providing our institution the opportunity to participate in the AQVH 2025.

REFERENCES

- [1] IBM Quantum. Qiskit documentation. Available: <https://quantum.cloud.ibm.com/docs>
- [2] Qiskit Textbook. Learn quantum computing with Python. Available: <https://qiskit.org/textbook/>
- [3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2010.
- [4] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018. <https://doi.org/10.22331/q-2018-08-06-79>
- [5] IBM Quantum Experience. Available: <https://quantum-computing.ibm.com/>
- [6] J. R. Johansson, P. D. Nation, and F. Nori, "QuTiP 2: A Python framework for the dynamics of open quantum systems," *Computer Physics Communications*, vol. 184, no. 4, pp. 1234–1240, 2013. <https://doi.org/10.1016/j.cpc.2012.11.019>
- [7] C. Gidney. Quirk: A drag-and-drop quantum circuit simulator. Available: <https://algassert.com/quirk>
- [8] Shadcn. Shadcn UI documentation. Available: <https://ui.shadcn.com/docs/installation/next>
- [9] Supabase. Supabase authentication guide. Available: <https://supabase.com/docs/guides/auth>
- [10] Supabase. Use Supabase Auth with Next.js. Available: <https://supabase.com/docs/guides/auth/quickstarts/nextjs>
- [11] Qiskit. Qiskit Aer density matrix simulation tutorial. Available: https://qiskit.github.io/qiskit-aer/tutorials/1_aersimulator.html
- [12] QuTiP. Visualization of quantum states and processes. Available: <https://qutip.org/docs/4.6/guide/guide-visualization.html>
- [13] M. Chauhan, "Visualizing quantum dynamics with QuTiP," Medium, 2023. Available: <https://medium.com/@mihir473/visualizing-quantum-dynamics-with-qutip-d5839bcd9c51>
- [14] M. M. Williams, "QCVIs: A quantum circuit visualization and education platform for novices," Harvard DASH, 2021. Available: <https://dash.harvard.edu/bitstreams/1f13af4-89ae-4997-a820-4b91831edb0a/download>
- [15] D. Brahmabhatt, "An open-source data storage and visualization platform for quantum computing research," *Scientific Reports*, vol. 14, no. 1, p. 72584, 2024. <https://doi.org/10.1038/s41598-024-72584-9>
- [16] Qiskit. Exact and noisy simulation with Qiskit Aer primitives. Available: <https://quantum.cloud.ibm.com/docs/guides/simulate-with-qiskit-aer>
- [17] QuTiP. Visualization and animation. Available: <https://qutip.readthedocs.io/en/latest/apidoc/visualization.html>
- [18] Shadcn. Next.js + Shadcn UI documentation. Available: <https://ui.shadcn.com/docs/react-19>
- [19] Supabase. Build a user management app with Next.js. Available: <https://supabase.com/docs/guides/getting-started/tutorials/with-nextjs>
- [20] Supabase. Setting up server-side auth for Next.js. Available: <https://supabase.com/docs/guides/auth/server-side/nextjs>