COM3610 Dissertation Project

An Educational Quantum Circuit Builder and Simulator

Lea Button

Supervisor: Phil McMinn

This report is submitted in partial fulfilment of the requirement
for the degree of BSc Computer Science by Lea Button.

May 15, 2024

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Lea Button

Abstract

Quantum computing is a rapidly emerging field that, due to its nature, allows computations to be completed millions of times faster than classical computers. While this statement generalises the advantage of quantum computing over classical computing, it is for this reason that quantum computing has the potential to revolutionise fields such as cryptography, optimisation, and chemistry. The caveat is that quantum systems are incredibly complex and challenging to experiment with. Therefore, the development of quantum circuit simulators has become an essential tool for researchers to study and experiment with quantum systems.

Many simulators exist; however, the learning curve can be extremely steep, particularly for those new to quantum computing concepts. The project aims to create a beginner-friendly quantum circuit simulator, utilising helpful visualisations to allow users to understand their simulations fully. This report seeks to complete a comprehensive literature survey, present a project plan, detail the design and implementation, evaluate the project, and discuss future work.

Acknowledgements

I would like to first thank my supervisor, Phil McMinn, for his support and guidance throughout this project. I would also like to thank The University of Sheffield for providing me with not only the opportunity to undertake this project, but also the continuous support throughout my degree that made me capable of completing this project to the level that I have. Finally, I would like to especially thank those who participated in the user survey, as their feedback was invaluable in evaluating the success of the project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The fundamental difference between quantum computing and classical computing is in their units of information. In classical computing, information is represented by bits, while in quantum computing, information is represented by qubits. Qubits can be a combination of 0 and 1, called a *superposition* of 0 and 1. In contrast, classical bits are restricted to just 0 or 1 (see Figure 1.1). Qubits can exist in superposition until being measured — measurement forces them to collapse into either 0 or 1.

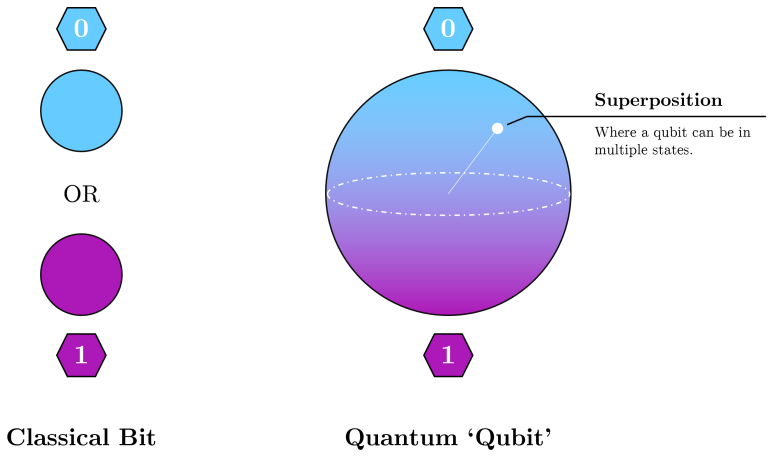


Figure 1.1: Visual comparison of a bit and a qubit (Adapted from [9][6]).

Entanglement is another unique property of qubits, allowing multiple qubits' states to depend on each other. These two properties allow quantum systems to represent and process exponentially more information in parallel than classical systems. This is what makes quantum computing faster than classical computing for specific applications. For example,

quantum computers can use Shor's Algorithm [14] to efficiently factor numbers, which is a problem believed to be hard for classical computers — the best-known algorithm for classical computers to factor numbers runs in sub-exponential time [43].

With the potential to revolutionise many industries, education and research in this field is imperative.

1.1 Aims and Objectives

There is an abundance of quantum computing educational materials tailored to those well-versed in physics, leaving a significant gap in catering to those with a computer science background. This project aims to develop an educational platform with a quantum circuit simulator that is intuitive and user-friendly and provides guided resources aimed at educating the user on quantum computing and programming. The platform will specifically target those with undergraduate-level knowledge. The resources will be focused on practically teaching quantum computing and algorithms, teaching through application rather than through rigorous theory and proofs [27].

One major constraint to the project is how resource-intensive quantum simulation can be, particularly for large circuits. Certain quantum hardware behaviours and characteristics can be challenging to simulate using classical hardware. For example, quantum systems can exist in a state of superposition, and as the size of the system grows, the number of states increases exponentially. This makes it increasingly computationally difficult to simulate. Limits may have to be set on circuit size and complexity to keep the user experience acceptable and the simulator practical.

1.2 Overview of the Report

Chapter 2: A comprehensive literature survey to provide foundational context for the project.

Chapter 3: Project requirements and analysis will be detailed in-depth, along with considerations on the methods of evaluation and testing that will be utilised throughout the project.

Chapter 4: Design decisions and justifications will be discussed, including the back-end, front-end, database and simulator design.

Chapter 5: Project implementation will be detailed, including the final technology stack, back-end and front-end implementation, deployment, and testing.

Chapter 6: The results of the project will be discussed, including the user survey feedback, limitations of the work and future developments.

Chapter 7: Project conclusions.

Chapter 2

Literature Survey

To design and develop a quantum circuit simulator, foundational knowledge of quantum computing concepts and quantum circuit simulation is required. This chapter will provide an overview of the concepts of quantum computing and quantum circuit simulation, as well as an overview of existing quantum circuit simulators and quantum simulation tools.

2.1 Quantum Computing

This section will draw parallels between classical computing and quantum computing to introduce the fundamental concepts of quantum computing, including qubits, quantum gates, and quantum circuits.

2.1.1 Qubits and Quantum States

We are familiar with bits from classical computing, but quantum computing uses a different unit of information called a qubit. A bit is physically represented by a system that can exist in two distinct states (on or off, 0 or 1). For example, a transistor that is either on or off. A (different) physical system also characterises a qubit. For example, an electron can represent a qubit in a superposition of spin-up and spin-down states.

Dirac Notation

Dirac notation [7] is a compact notation for quantum mechanics that uses angle brackets $\langle\rangle$ and bars $|$. Intuitively, Dirac notation is also called *Bra-ket*[1] notation. A bra is the form $\langle f|$ which represents a linear map/transformation $\mathbf{f}$. A ket is the form $|v\rangle$ which represents a vector $\mathbf{v}$. Applying a linear map $\mathbf{f}$ to a vector $\mathbf{v}$ is detonated as $\langle f|v\rangle$

The two distinct states, 0 or 1, are also known as the *computational basis states*. A qubit in either of the computational basis states can be represented as a column vector, or, under Dirac notation, as a ket:

[1] Specifically Bra-ket, not *bracket*.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Throughout this report, while column vectors could be used, Dirac notation will be used to detonate the state of a qubit.

Visualisation

It can be challenging to conceptualise the state of a qubit — we can understand a bit's state in a number of ways, such as a voltage level, a transistor's state, or abstractly as a light switch. A qubit, however, is not as simple. A qubit can exist as a 'combination' of both states at once. There are an infinite number of possible 'combination' states a qubit can be in, so an 'ON/OFF' analogy is not sufficient. A qubit's state can be visualised on a sphere of radius 1 called a Bloch sphere, as shown in Figure 2.1. The distinct $|0\rangle$ and $|1\rangle$ states can be seen as the north and south poles of the Bloch sphere, respectively [43].

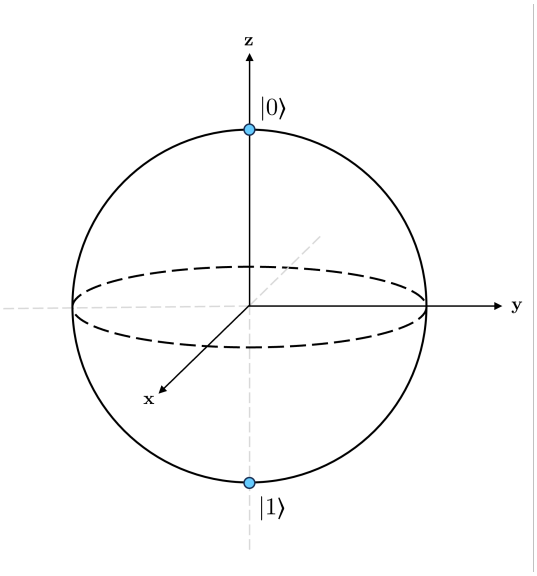


Figure 2.1: Bloch sphere representation of a qubit (Adapted from [43][28]).

$|0\rangle$ corresponds to the (0,0,1) point on the sphere, and $|1\rangle$ to (0,0,-1). With this system, any point on the surface of the Bloch sphere represents a valid state of a qubit. This visualisation can help us represent any state of a qubit, including superpositions.

Superposition

In contrast to a bit in classical computing, while a qubit may be in one of two distinct states, a qubit can also be in a superposition of both these states at once. The superposition of both states is a linear combination of the *computational basis states*, $|0\rangle$ and $|1\rangle$. Superposition is denoted as:

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Where α and β are complex numbers representing the probability of measuring the qubit in the $|0\rangle$ and $|1\rangle$ states, respectively. Consider the following example:

$$|\phi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

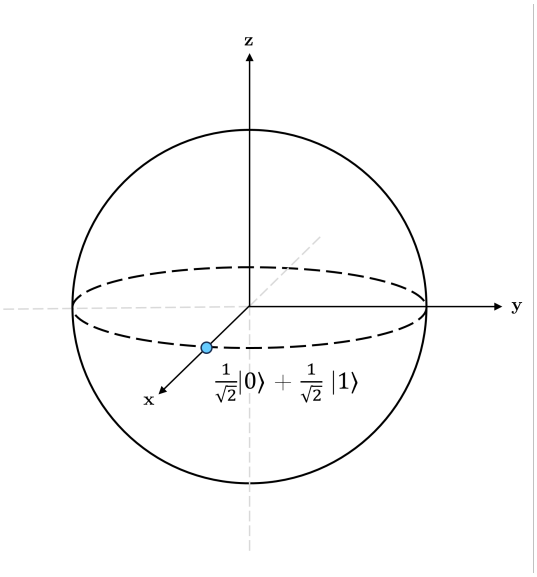


Figure 2.2: Bloch sphere representation of $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ (Adapted from [43][28]).

This is a superposition of $|0\rangle$ and $|1\rangle$ with equal probability. Intuitively, on a Bloch sphere, this point lies on the equator exactly between the north and south poles, as shown in Figure 2.2. This specific state appears frequently and is called the *plus state*: $|+\rangle$. Similarly, there exists the *minus state*, *i state*, and *minus i state*.

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$|i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$$

$$|-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$$

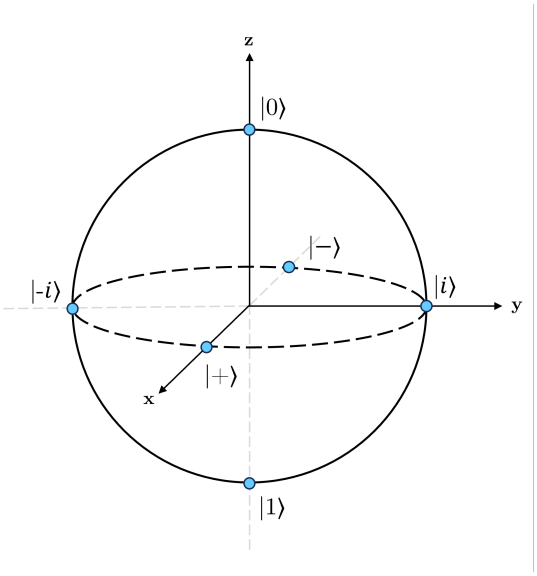


Figure 2.3: Bloch sphere representation of the plus, minus, i and minus i states (Adapted from [43][28]).

Measurement

In classical computing, a bit can be measured in a state of either 0 or 1. On the other hand, in quantum computing, although a qubit can *exist* in a superposition of both states, it can only be *measured* to be a single, definite value rather than a superposition. Measurement forces the qubit to collapse into one of the two computational basis states.

The probability of measuring a qubit in a particular state is the square of the magnitude of the coefficient of that state in the qubit's superposition (the norm-square). For example ($|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$), the probability of measuring a qubit in the $|0\rangle$ state is $|\alpha|^2$, and the probability of measuring a qubit in the $|1\rangle$ state is $|\beta|^2$.

In the previous example (Figure 2.2), the probability of measuring the qubit in the $|0\rangle$ state is $|\frac{1}{\sqrt{2}}|^2$, and the probability of measuring the qubit in the $|1\rangle$ state is also $|\frac{1}{\sqrt{2}}|^2$ [28]. Therefore, the probability of measuring the qubit in *either* state is 1 ($|\frac{1}{\sqrt{2}}|^2 + |\frac{1}{\sqrt{2}}|^2$). A state is considered *normalised* if the sum of the squares of the magnitudes of α and β equals 1.

Tensor Product

Before moving on to multiple qubits, it is important to introduce the *tensor product*, detonated by $\otimes$. The tensor product is a way of combining two or more qubits into a single system.

For example (using Dirac notation):

$$|0\rangle \otimes |0\rangle = |00\rangle$$

Referring back to states as column vectors:

$$\begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\begin{bmatrix}1\\0\end{bmatrix}\\0\begin{bmatrix}1\\0\end{bmatrix}\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

Multiple Qubits

A classical computer can have multiple bits; similarly, a quantum computer can have multiple qubits. With two bits, there would be four possible states: 00, 01, 10, and 11. Similarly, two qubits have four computational basis states: $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. These states are a *tensor product* of the individual qubits' states. For example, the state $|01\rangle$ is a tensor product of the $|0\rangle$ state of the first qubit and the $|1\rangle$ state of the second qubit. A pair of qubits can be in a superposition of these four states:

$$|\phi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

Where α_{00}, α_{01}, α_{10} and α_{11} are complex numbers representing the probability of measuring the qubit in the $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$ states, respectively. These states are sometimes written as decimal numbers, $|0\rangle$, $|1\rangle$, $|2\rangle$ and $|3\rangle$. The conventional order in which the qubits are written is often disputed [5]. However, *little endian* will be used throughout this survey, where the least significant bit is written first.

To generalise this, the state of n-qubits is a superposition of 2^n computational basis states:

$$|\phi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$$

Where α_i is the probability amplitude of the i^{th} computational basis state.

Measurement with Multiple Qubits

If we want to measure an individual qubit in a multi-qubit system, we want the probabilities of that single qubit collapsing into the $|0\rangle$ and $|1\rangle$ states. For example [43]:

$$|\phi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{2}|01\rangle + \frac{\sqrt{3}}{4}|10\rangle + \frac{1}{4}|11\rangle$$

The probability of measuring the left qubit in the $|0\rangle$ state is the sum of the norm-squares of the coefficients of $|00\rangle$ and $|01\rangle$:

$$|\frac{1}{\sqrt{2}}|^2 + |\frac{1}{2}|^2 = \frac{3}{4}$$

The probability of measuring the left qubit in the $|1\rangle$ state is the sum of the norm-squares of the coefficients of $|10\rangle$ and $|11\rangle$:

$$|\frac{\sqrt{3}}{4}|^2 + |\frac{1}{4}|^2 = \frac{1}{4}$$

After this measurement, the state must collapse into one of the two states. If the left qubit is measured to be in the $|0\rangle$ state, the state of the system is now:

$$|\phi\rangle = \frac{\sqrt{6}}{3}|00\rangle + \frac{\sqrt{3}}{3}|01\rangle$$

Similarly, if the left qubit is measured to be in the $|1\rangle$ state, the state of the system is now:

$$|\phi\rangle = \frac{\sqrt{3}}{2}|10\rangle + \frac{1}{2}|11\rangle$$

The same logic can be applied to measure the second qubit. For example, if the left qubit was measured to be in the $|0\rangle$ state, the probability of measuring the second qubit in the $|0\rangle$ would be:

$$|\frac{\sqrt{6}}{3}|^2 = \frac{2}{3}$$

If we calculate this for all possible outcomes, we get the same probabilities as if we measured the entire system at once:

$$|00\rangle = \frac{3}{4} \times \frac{2}{3} = \frac{1}{2}$$

$$|01\rangle = \frac{3}{4} \times \frac{1}{3} = \frac{1}{4}$$

$$|10\rangle = \frac{1}{4} \times \frac{3}{4} = \frac{3}{16}$$

$$|11\rangle = \frac{1}{4} \times \frac{1}{4} = \frac{1}{16}$$

Entanglement

Consider the following example:

$$|\phi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Notice there is no possibility of measuring either $|01\rangle$ or $|10\rangle$, and if either qubit is measured individually, the other qubit's state can be deduced without measurement. This is called *entanglement*. Entanglement is a phenomenon where the state of one qubit is correlated to the state of another qubit.

2.1.2 Quantum Gates and Circuits

Just as classical computers use logic gates to manipulate bits, quantum computers use quantum gates to manipulate qubits. This sub-section will introduce some of the most common quantum gates.

Single-Qubit Gates

Single-qubit gates are quantum gates that operate on a single qubit. There are properties required for a gate to be considered valid:

1. The total probability across superpositions must remain 1.
2. The gate must be a linear operator.
3. The gate must be reversible.
4. The gate must be unitary.

Identity Gate The identity gate is the simplest quantum gate. It does nothing to the qubit and is represented by the identity matrix:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$I|0\rangle = |0\rangle$$

$$I|1\rangle = |1\rangle$$

Pauli-X Gate The Pauli-X gate is the quantum equivalent of the classical NOT gate:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$X|0\rangle = |1\rangle$$

$$X|1\rangle = |0\rangle$$

The Pauli-X gate is also known as the *bit-flip* gate and can be imagined as a 180° rotation around the x-axis of the Bloch sphere.

Pauli-Y Gate The Pauli-Y gate is similar to the Pauli-X gate, but it is a 180° rotation around the y-axis of the Bloch sphere:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

$$Y|0\rangle = i|1\rangle$$

$$Y|1\rangle = -i|0\rangle$$

Pauli-Z Gate The Pauli-Z gate is a 180° rotation around the z-axis of the Bloch sphere:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$Z|0\rangle = |0\rangle$$

$$Z|1\rangle = -|1\rangle$$

Phase Gate The phase gate is the square root of the Pauli-Z gate, and is a 90° rotation around the z-axis of the Bloch sphere:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

$$S|0\rangle = |0\rangle$$

$$S|1\rangle = i|1\rangle$$

T Gate The T gate is the fourth root of the Pauli-Z gate, and is a 45° rotation around the z-axis of the Bloch sphere:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}$$

$$T|0\rangle = |0\rangle$$

$$T|1\rangle = e^{\frac{i\pi}{4}}|1\rangle$$

Hadamard Gate The Hadamard gate is a 180° rotation around the x-axis of the Bloch sphere, followed by a 90° rotation around the y-axis of the Bloch sphere:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H|0\rangle = |+\rangle$$

$$H|1\rangle = |-\rangle$$

These are the most common single-qubit gates. However, any unitary 2x2 matrix can be used as a single-qubit gate.

Two-Qubit Gates

Two-qubit gates are quantum gates that operate on two qubits. Similar to single-qubit gates, some standard gates are used in quantum computing.

Controlled-NOT Gate The controlled-NOT gate, or CNOT gate, is a two-qubit *'quantum XOR'* gate. It is sometimes referred to as the CX or controlled-X gate. It performs a NOT operation on the second qubit if the first qubit is in the $|1\rangle$ state:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$CNOT|00\rangle = |00\rangle$$

$$CNOT|01\rangle = |01\rangle$$

$$CNOT|10\rangle = |11\rangle$$

$$CNOT|11\rangle = |10\rangle$$

There are many other two-qubit gates, such as the controlled X, Y, and Z gates and the controlled Hadamard gate. These follow the same concept as the CNOT gate but perform different operations on the second qubit depending on the state of the first qubit.

Circuit Symbols

Table 2.1 shows the circuit symbols for the quantum gates discussed in this section.

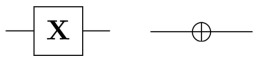
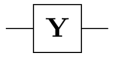
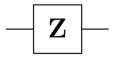
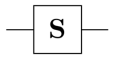
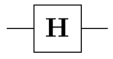
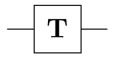
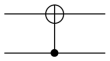
Gate	Circuit Symbol(s)
Pauli-X	
Pauli-Y	
Pauli-Z	
Phase Gate	
Hadamard Gate	
T Gate	
CNOT Gate	

Table 2.1: Quantum gates and their circuit symbols.

Quantum Circuits and Combining Gates

We can construct quantum circuits by combining quantum gates. For example, the following circuit applies a Hadamard gate to a single qubit, followed by a Pauli-Z gate:

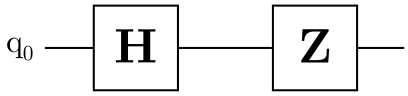


Figure 2.4: Quantum circuit showing a Hadamard gate followed by a Pauli-Z gate.

The circuit can be represented as a matrix by multiplying the matrices of the gates:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \times \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

Any sequence of gates can be combined in this way and then treated as a single gate U in calculations.

2.1.3 Quantum Error Correction

Quantum error correction involves methods to protect quantum information from errors caused by decoherence and other quantum noise. Quantum error correction is essential for

quantum computing because of the fragile nature of quantum systems. It can be challenging to maintain the state of a qubit for a long enough time to perform a computation, and due to the no-cloning theorem [44], quantum information cannot be duplicated in the same way as classical information.

In classical computing, a bit-flip error is when a bit is flipped from 0 to 1 or 1 to 0. With quantum computing, this concept is more complex — there are as many possible superpositions of states as there are positions on/in the Bloch sphere. Qubits can experience *partial bit-flip* errors and *partial phase-flip* errors [35].

Bit-Flip

A *bit-flip* error is a rotation around the x-axis of the Bloch sphere. We can correct these errors using a three-qubit scheme called the *bit-flip code*, designed to protect against a single bit-flip error. The scheme uses three qubits to encode one qubit of information; for the same reason that three bits are needed for the classical equivalent - a 'parity vote' can be taken. The three qubits are called a *qutrit*. The qutrit is encoded in the following way:

$$|0_L\rangle \rightarrow |000\rangle, \quad |1_L\rangle \rightarrow |111\rangle$$

Where $|0_L\rangle$ and $|1_L\rangle$ detonate the logical qubit being represented[2]. A superposition is represented by the following:

$$|\psi\rangle = \alpha|0_L\rangle + \beta|1_L\rangle$$

$$|\psi\rangle = \alpha|000\rangle + \beta|111\rangle$$

The parities of the logical qubit are calculated using two other qubits in the $|0\rangle$ state using CNOT gates. This pair of qubits is referred to as the *syndrome*, which is used to help detect and correct errors. Figure 2.5 shows the circuit for the bit-flip code.

[2]Rather than two separate examples of encoding a qubit.

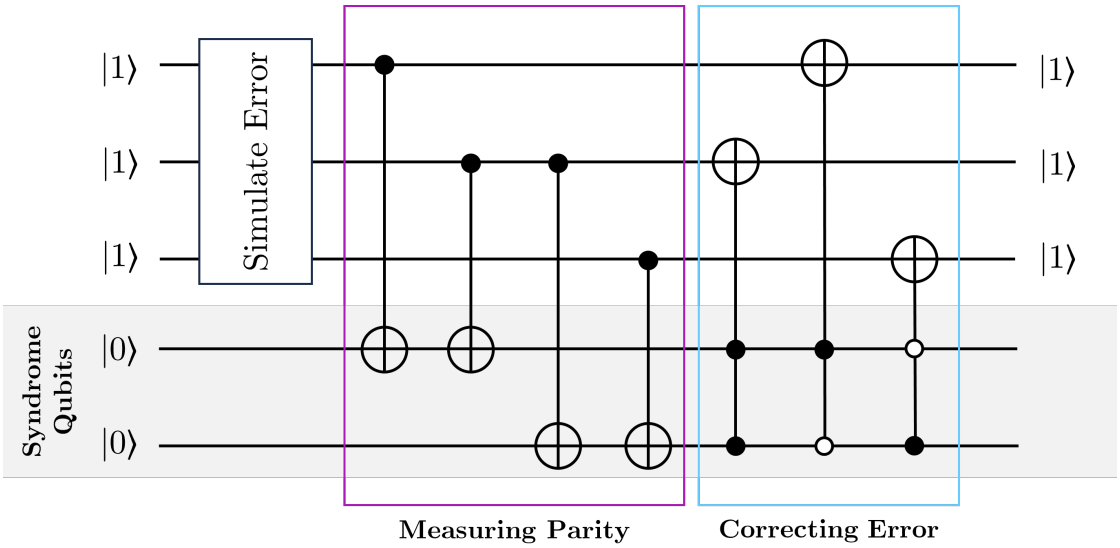


Figure 2.5: Quantum circuit showing the bit-flip circuit (Adapted from [23, p. 6][13, p. 122]).

Phase-Flip

A *phase-flip* error is a rotation around the z-axis of the Bloch sphere. Similar to the bit-flip code, we can use a three-qubit scheme called the *phase-flip code* to correct these errors. The phase-flip code also uses three qubits to encode one qubit of information. The qutrit is encoded in the following way:

$$|0_L\rangle \rightarrow |++\!+\rangle, \quad |1_L\rangle \rightarrow |--\!-\rangle$$

A superposition is represented by the following:

$$|\psi\rangle = \alpha|0_L\rangle + \beta|1_L\rangle$$

$$|\psi\rangle = \alpha|++\!+\rangle + \beta|--\!-\rangle$$

Similar to the bit-flip code, the parities of the logical qubit are calculated using two other qubits in the $|0\rangle$ state using CNOT gates. In contrast to the bit-flip code, Hadamard gates are applied to the logical qubit before measuring their parity and after correcting the error. Applying the Hadamard gates allows us to work with the same encoding as the bit-flip code. Figure 2.6 shows the circuit for the phase-flip code.

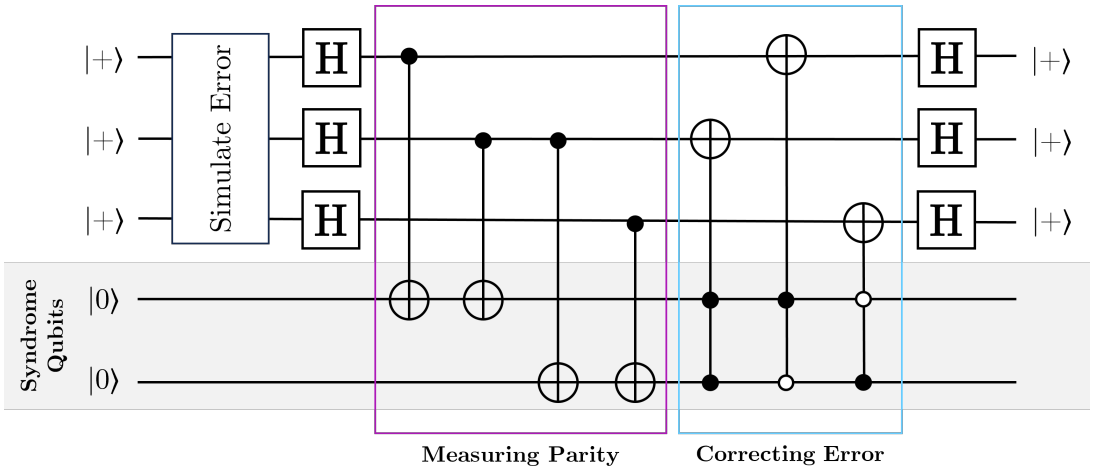


Figure 2.6: Quantum circuit showing the phase-flip circuit (Adapted from [13, p. 122][40]).

Shor Code

Peter Shor proposed this quantum error correction scheme in 1995 [37]. The Shor code is a nine-qubit scheme that can correct any single-qubit error. It is a combination of the bit-flip and phase-flip codes. Starting with the phase-flip code:

$$|\psi\rangle = \alpha|0_L\rangle + \beta|1_L\rangle \tag{2.1}$$

$$|\psi\rangle = \alpha|{+}{+}{+}\rangle + \beta|{-}{-}{-}\rangle \tag{2.2}$$

$$|\psi\rangle = \frac{\alpha}{2\sqrt{2}}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle) + \beta|{-}{-}{-}\rangle \quad \text{Where } |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{2.3}$$

$$|\psi\rangle = \frac{\alpha}{2\sqrt{2}}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle) + \frac{\beta}{2\sqrt{2}}(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle) \quad \text{Where } |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.4}$$

Then, for the bit-flip code, we replace $|0\rangle$ and $|1\rangle$ with the $|000\rangle$ and $|111\rangle$ encodings, respectively:

$$|\psi\rangle = \frac{\alpha}{2\sqrt{2}}(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) + \frac{\beta}{2\sqrt{2}}(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \tag{2.5}$$

Summary

Quantum error correction is an important consideration that should be made when programming quantum algorithms. If the algorithm is to be run on a real system, it will not be transferrable from a simulation without quantum error correction. Equally, when

designing a quantum circuit simulator, it may be desirable to simulate quantum noise and errors for this reason.

2.2 Simulation

Quantum simulation is the process of using a classical computer to simulate a quantum system's behaviour. This section will introduce existing quantum circuit simulators and tools for simulation.

2.2.1 Existing Implementations of Quantum Circuit Simulators

There are many existing quantum circuit simulators, such as *Quirk* [16] and IBM's *Quantum Composer* [20]. Both are widely used and have a graphical user interface.

Quirk

Quirk is a user-friendly drag-and-drop quantum circuit simulator written in JavaScript. It is browser-based and has several key features:

- Ability to create circuits of up to 16 qubits.
- Ability to create custom gates.
- Ability to export circuits to JSON, HTML, or as a link.
- Extra information is displayed when hovering over gates, such as the gate's matrix and a visualisation of the rotation it performs.
- A small selection of example circuits, including *Shor Period Finding* and *Grover's Search*.

Figure 2.7 shows the GUI of Quirk with an example of a circuit demonstrating *bit-flip error correction* created using Quirk.

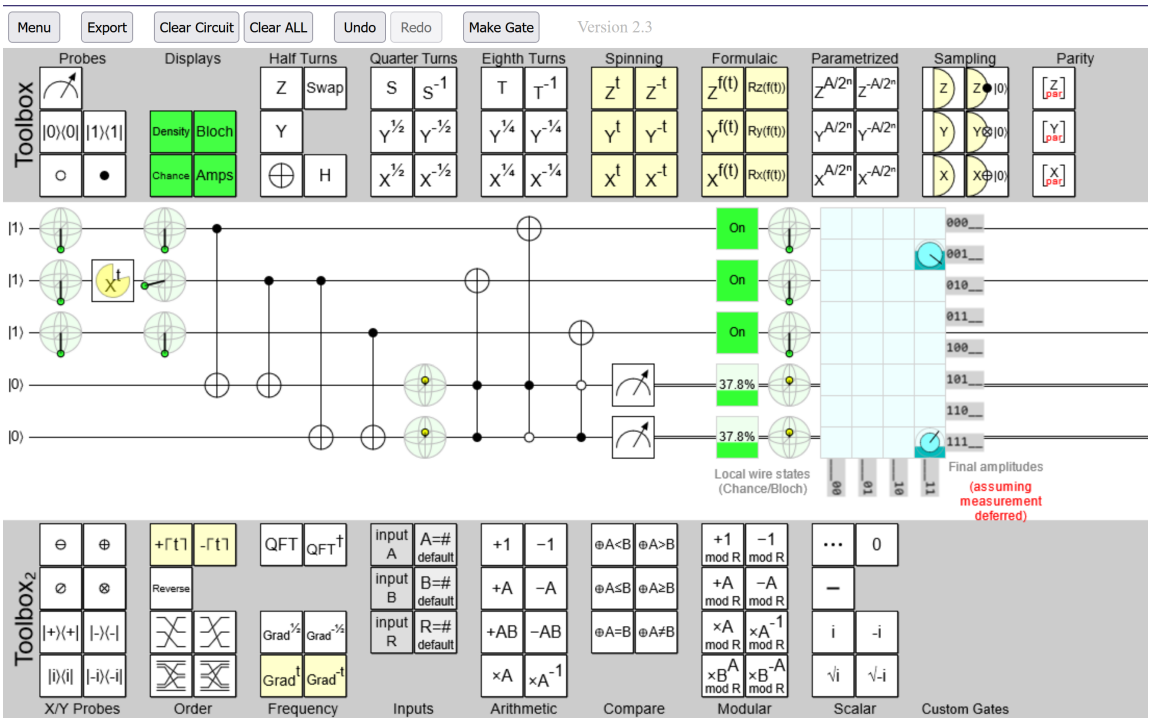


Figure 2.7: Quirk GUI.

Quirk, however, does not feature a quantum compiler or a QPU (Quantum Processing Unit) back-end [11]. It is purely a simulator with no functionality to run circuits on a real quantum computer.

IBM Quantum Composer

The IBM Quantum Composer is another drag-and-drop quantum circuit simulator. It is browser-based and has several key features:

- Ability to simulate circuits of up to 16 qubits. However, larger circuits can be created and run on IBM's QPU back-end.
- Ability to save circuits to an IBM Quantum account.
- Ability to download circuits as images.
- Extra information is available when right-clicking elements.
- Generates Qiskit and QASM code, allowing code input in both QASM.

Figure 2.8 shows the GUI of IBM's Circuit Composer with an example of a circuit demonstrating *bit-flip error correction* in the simulator.

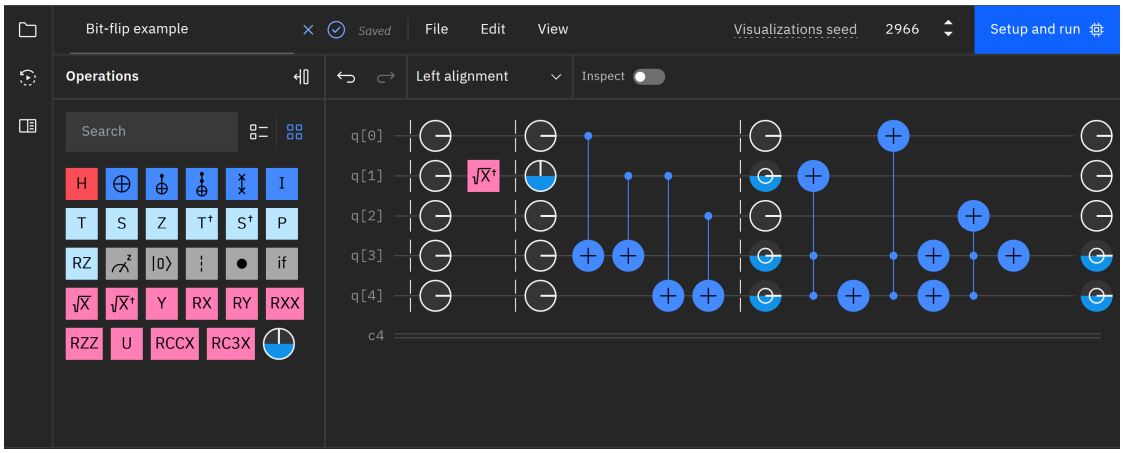


Figure 2.8: IBM Quantum Composer's GUI.

2.2.2 Quantum Simulation Tools

Many tools to aid in quantum circuit simulation exist[3] — IBM's *Qiskit* [32], Google's *Cirq* [4], and Microsoft's *Quantum Development Kit (QDK)* [24] each provide full quantum simulation environments and many other features. All these tools boast thorough documentation, tutorials, and a large community of users.

- **Qiskit** is an open-source quantum software development kit (SDK) based on Python. Qiskit provides tools for quantum simulation and has freely accessible QPUs.
- **Microsoft's QDK** is an open-source quantum computing software development kit (SDK). It is based on C# and provides tools for quantum simulation. QDK also provides a QPU back-end, but it is not freely accessible.
- **Cirq** is an open-source Python framework providing tools for quantum simulation, designed specifically for *Noisy Intermediate-Scale Quantum* (NISQ). Although Cirq does not provide a QPU back-end for free, it does provide a simulator and a compiler.

These tools and simulators will be discussed and analysed in more detail in Section 3.1.

2.3 Teaching Quantum Computing

Quantum computing is a complex subject, and teaching it can be challenging. This section will review effective approaches to teaching quantum computing to a computer science audience.

[3]The differentiation between applications such as Quirk and these tools is that the former are interactive applications, while the latter are programming frameworks.

2.3.1 Catering to a Computer Science Audience

Research suggests that, especially when catering to those without a background in quantum physics, interactively teaching quantum concepts is more engaging and accessible.

By teaching through programming and focusing on practical aspects of quantum algorithms, one can learn the fundamentals of quantum computing without needing to understand the underlying physics. In a report titled *Teaching Quantum Computing through a Practical Software-driven Approach* by *Mariia Mykhailova* and *Kyrsta M. Svore* [27], this idea is consolidated. The report is based on a course taught at the University of Washington, where undergraduate students were taught quantum computing through programming. The course was taught using Microsoft's QDK, and students were required to complete a series of programming assignments. The course was successful, and the authors concluded that a *software-orientated* approach effectively teaches the practical aspect of quantum computing to a computer science audience.

To further this, a course designed for undergraduate ICT students [3], which was taught using Qiskit and IBM's Circuit Composer, also resulted in positive feedback from the students involved in the course. The Circuit Composer was used as a step between learning initial theory and programming using Qiskit.

Another paper titled *Teaching Quantum Computing with the QuIDE Simulator* [36] outlined the importance of using QuIDE [22] for the course being taught. The paper's authors developed QuIDE specifically to value their educational requirements and cater to their students' needs. The paper also featured a survey comparing QuIDE to *libquantum* [2]. The survey conducted as part of the paper concluded that QuIDE was more user-friendly and easier to use than *libquantum*[4].

Overall, success has been found using a quantum circuit simulator to aid in teaching quantum computing. Studies show that students with a computing background are more engaged and find it easier to learn quantum computing through practical exercises to support the theory being taught.

2.4 Summary

This literature survey has introduced the background knowledge required to understand, design, and develop an educational quantum circuit simulator. Existing simulators have been introduced, and the most effective approaches to teaching quantum computing have been discussed. A *software-driven* [27] approach has been shown to be particularly effective in educating a computing audience and helps reinforce the importance of quantum circuit simulators for teaching. The next chapter will discuss the design of the quantum circuit simulator and elaborate further on the direction of this project.

[4]To clarify, the paper reinforces the fact that a visual 'drag-and-drop'-style environment (QuIDE) is more effective for learning than purely programming (libquantum).

Chapter 3

Requirements and Analysis

3.1 Technology Analysis

3.1.1 Existing Circuit Simulators

There are many existing circuit simulators that are available for use. Many of these tools are open-source, free to use, and offer a wide range of features. Reviewing the capabilities of these tools has played a significant part in choosing the direction of this project, pushing toward a simulator designed for educational use.

Quirk and IBM's Circuit Composer

Quirk [16] is one of the earliest quantum circuit simulators available, popularised by its simplicity. It is a web-based tool that allows users to create and simulate quantum circuits in a drag-and-drop interface. IBM's Circuit Composer [20] is newer than Quirk, but offers a similar experience. It is also a web-based tool that allows users to create and simulate quantum circuits in a drag-and-drop interface. It has a more modern design than Quirk, and offers more features, such as the ability to run the circuit on a real quantum computer.

Looking at both these tools alone, along with consideration for the other tools available, it is clear that there is no shortage of quantum circuit simulators. This is what led to the decision to create a simulator that is designed for educational use rather than a general-purpose simulator. Given the time constraints of this project, it would be difficult to create a simulator that is as feature-rich as the existing tools. However, minimising the features and focusing on the educational aspect of the simulator will allow for a more in-depth and useful tool to be created. This decision is justified by the fact that many additional features included in the existing tools are not necessary for educational use. This project will aim to create a simulator that is as simple to use as possible, without sacrificing the core features that are necessary for a quantum circuit simulator.

3.1.2 Comparison of Quantum Simulation Tools

There are a number of quantum simulation tools that could be used for the back-end of the simulator. Table 3.1 shows a comparison between Qiskit [32], Cirq [4], and QDK [24]. All three are open-source, free to use, and have good documentation and learning resources available. Qiskit and Cirq are both written for Python, while QDK is written for Q#. Naturally, Qiskit and Cirq have an advantage over QDK for this project as they are written in Python, an already familiar language. Qiskit and Cirq both have good documentation and many features available. However, Cirq is specifically designed for quantum circuit development, while Qiskit is more general-purpose. Therefore, **Cirq** is the most suitable tool for the back-end of the simulator. While Qiskit is more established than Cirq, Cirq is still a well-established tool with all the features available that the project may require.

	Qiskit	**QDK**	**Cirq**
Programming Language	Python, C++	Q#	Python
License	Apache 2.0	MIT	Apache 2.0
Advantages	Has many features, with very good documentation and community support.	Has many features, with good documentation.	Specifically designed for quantum circuit development, and promotes low level control over gates.
Disadvantages		Limited support for non-.NET languages, creating a learning curve to understand Q#.	

Table 3.1: Comparison of different quantum simulation tools.

3.1.3 Comparison of Front-end tools

The suitability of existing front-end frameworks and tools for both website and desktop applications will influence whether the project will be a website or desktop application.

There are a number of front-end tools that could be used to create a desktop application. Table 3.2 shows a comparison between PyQt [34], JavaFX [29], and Tkinter [31]. Specifically for this project, PyQt and JavaFX have a significant advantage over Tkinter in that both have GUI builders, allowing time to be saved in the development process. The aesthetics of the GUI are important for making the simulator intuitive and easy to use; however, using a GUI builder rather than building the GUI from scratch will allow more time to be spent on the core functionality of the simulator without taking away from the ability to create a functional and simple GUI. While not having a GUI builder, Tkinter has the smallest learning curve of the three tools and is the most lightweight.

Considering Cirq will be used in the back-end of the simulator, it would be beneficial to use a front-end tool that is also written in Python. This would allow for a more seamless integration between the front-end and back-end of the simulator. With this in mind, **PyQt** is the most suitable tool for creating a desktop application — despite having a steeper learning curve than Tkinter, it has good documentation and learning resources available.

	PyQt	**JavaFX**	**Tkinter**
Programming Language	Python	Java	Python
License	GPL	GPL	Python License
Advantages	Qt Designer will simplify the design process and PyQt has many features.	Gluon will simplify the design process, and JavaFX has many features.	Easy to use and good for smaller, lightweight applications
Disadvantages	Bigger learning curve.	Big learning curve and may not integrate well with a Python backend.	Limited features and less aesthetically pleasing than other frameworks.

Table 3.2: Comparison of different front-end desktop tools.

Alternatively, the project could be a website. Table 3.3 shows a comparison between React [10], Vue [41], and Angular [17]. All three are JavaScript-based, have the same licensing, and have available GUI builders. The application that will be created for this project will be relatively small, so Vue or Angular would be more suitable than React, which is generally used for larger applications. Vue has a smaller learning curve than Angular and is more lightweight. Angular has also been deprecated, so **Vue** is the most suitable tool for creating a website application.

	React.js	**Vue.js**	**Angular.js**
Programming Language	JavaScript	JavaScript	JavaScript
License	MIT	MIT	MIT
Advantages	Promotes reusable components, and lots of learning tutorials available.	Simple and suited to smaller applications.	Promotes reusable components.
Disadvantages	Large learning curve, especially due to the usage of JSX. Better suited to large and complex applications	There is a language barrier with a lot of forum discussions, plugins, and instructions.	Large learning curve, with needing to understand TypeScript. No longer receives official support.

Table 3.3: Comparison of different front-end website tools.

Given the choice between a website or a desktop application, there are several factors to consider. A website application would be more accessible than a desktop application, as it would be available to anyone with an internet connection. A website application would also easily enable cross-compatibility by nature. From a user's perspective, a website application would be more convenient as it would not require any installation. Particularly for educational purposes, the ability to quickly access the application would be beneficial. However, a desktop application would allow for offline usage and for the application to take advantage of the user's local resources for more robust computations.

Both desktop and website applications have their advantages and disadvantages, but the decision ultimately comes down to the purpose of the application. For this project, the application will be designed for educational purposes. Therefore, the accessibility and convenience of a website application are more important than the ability to use local resources.

Therefore, a **website application** is the most optimal choice for this project. If time permits, a hybrid application may be created from the website application, allowing the application to be used offline and to take advantage of local resources. However, this will not be a priority for the project.

3.1.4 Comparison of Back-end tools

There are several back-end tools that could be used to create a website application, however, it is important to choose a tool that will fit this project's scope. Table 3.4 shows a comparison between Django [8], Flask [12], and Ruby on Rails [33]. For this project, Django or Flask would be more suitable than Ruby on Rails, as both are written in Python which would support the decision to use Cirq.

	Django	Flask	Rails
Programming Language	Python	Python	Ruby
License	BSD 3-Clause	BSD 3-Clause	MIT
Advantages	Full-featured and expandable, suitable for rapid development.	Lightweight, provides more control over design.	Optimised for fast prototyping.
Disadvantages	Slowest of the three.	Can be complex when scaling up.	Can be inflexible at times.

Table 3.4: Comparison of different back-end website tools.

Considering the project will have a short development time, Django would be more suitable than Flask as it has more features available out-of-the-box, and supports a more rapid development process.

3.1.5 Chosen Technology Stack

To summarise the technology analysis, the back-end of the simulator will use Django with Cirq, and the front-end will use Vue. Additional libraries will be used to provide additional functionality, such as *NumPy* for linear algebra and *Matplotlib* for plotting and visualisation.

3.2 Project Requirements

The table shown in Table 3.5 summarises the requirements for the project. Each requirement has an indication of priority and an effort estimate. All *'must do'* requirements are needed to provide the core functionality; without these, simple simulation is impossible. The *'should do'* requirements are needed to produce a useful application for educational purposes and to create an intuitive circuit builder. The *'could do'* requirements are not necessary for the project but would be nice to have if time allows.

At its core, the project should be a simple quantum circuit simulator. The application should provide a simple and intuitive GUI that facilitates the creation and simulation of quantum circuits. Circuits should be created through a 'drag-and-drop' style interface, where gates can be dragged from a toolbar onto the circuit. The application must then be able to display the states of the qubit(s) after the circuit has been simulated. At a minimum, the simulator must be able to simulate around 6-8 input qubits, and the simulator should have presets of the most common gates that can be dragged onto the circuit:

- Pauli-X, Y, and Z gates.
- Phase gates.

- T gates.
- Hadamard gates.
- CNOT gates.

The application should also provide educational features to aid in the learning of quantum computing. These features should be simple and intuitive to use and are necessary for the goals of the project. These features include:

- The application must provide a simple tutorial to introduce the user to the simulator.
- The application must provide explanations of the gates and their effects.
- The application should provide a way to visualise the state of qubit(s) after the circuit has been simulated.
- The application should contain a series of interactive lessons that teach the user about quantum computing and quantum circuits.

Additional requirements for the project involve features that are not necessary to produce a satisfactory simulator, but would be useful to have if time allows. These include:

- The ability to save a circuit to a file and load circuits from files.
- The ability to save a composite gate — a group of gates that can be saved as a single gate and dragged onto the circuit.
- Multi-qubit preset gates, such as the Toffoli gate[1].
- The application could provide example circuits that the user can load and simulate.

[1] A common 3-qubit gate, sometimes referred to as a CCNOT gate.

ID	Requirement	Priority	Effort
R1	The application must contain a series of interactive lessons that teach the user about quantum computing fundamentals.	Must	High
R2	The application must be able to simulate two-qubit gates.	Must	High
R3	The application must be able to simulate single-qubit gates.	Must	High
R4	The application must allow multiple gates to be combined in parallel, series and or both.	Must	High
R5	The application must provide the user with a way to visualise qubit states.	Must	Medium
R6	The application must provide the user with useful information about simple gates and the state of a selected qubit.	Must	Medium
R7	The application must contain a walkthrough tutorial to introduce the user to the environment and indicate how to use the application.	Must	Medium
R8	The application should be able to simulate multi-qubit gates.	Should	High
R9	The application should allow a collection of simple gates to be compiled into a composite gate.	Should	Medium
R10	A user should be able to save and 're-use' a created composite gate.	Should	Medium
R11	The application should provide the user with exemplar circuits of quantum algorithms.	Should	Medium
R12	A user should be able to save a circuit to a file to allow circuits to be saved, shared, and modified.	Should	Low
R13	The application could contain an additional series of interative lessons that teach the user about more advanced concepts.	Could	High
R14	The application could be containerised to be deployed as a desktop application as well as a website application.	Could	High

Table 3.5: Project requirements.

3.3 Project Breakdown

The requirements of the project can be grouped into three main categories, in order of category priority:

- Circuit building and simulation:
  - R2 (must).
  - R3 (must).
  - R4 (must).
  - R5 (must).
  - R6 (must).
  - R8 (should).
  - R9 (should).
- Educational content:
  - R1 (must).
  - R7 (must).
  - R11 (should).
  - R13 (could).
- Users:
  - R10 (should).
  - R12 (should).

Here, the final requirement (R14) is standalone and is the last requirement that will be planned for implementation. Each priority category breaks the above list into smaller groups, which will form the basis of each planned iteration for the development of the project (Table 3.6).

Iteration	**1**	**2**	**3**	**4**	**5**
Priority	Must	Must	Should	Should	Could
Requirements	R2	R1	R8	R10	R14
	R3	R7	R9	R12	
	R4		R11	R13	
	R5				
	R6				

Table 3.6: Table showing each planned development iteration and involved requirements.

3.3.1 Evaluating and Measuring Success

The success of the project will be measured in a number of ways, most obviously by the extent to which the requirements are met. However, the success of the simulator will also come from its performance and the usability of the application for educational purposes.

While performance is not necessarily a priority for this project, it is still important to ensure that the simulator performs satisfactorily, where the user experience is not hindered by slow simulation times. The performance of the simulator will be measured by:

- The time it takes to simulate specific pre-defined circuits.
- How simulation time scales with circuit complexity.
- Resource usage of the simulator.
- Maximum number of qubits that can be simulated within a maximum time and resource limit.

This can be compared to previous iterations of this project to indicate progress made during development. It could also be compared to existing simulators [15] to provide a comparison between this project and baseline performance metrics. While not expected to be optimised for performance like other more established simulators, a comparison will provide an indication of the performance of the simulator given the expectations of the project.

The application's usability for educational purposes is more arbitrary and difficult to measure. However, it could potentially be measured by a survey of users, asking them to rate the usability of the application and asking them to compare it to other simulators they may have used.

Testing will be conducted throughout development to ensure the application is working as expected. The application's test suite will be written alongside the application itself. The test suite should be written using the *pytest* [30] framework, which is a popular testing framework for Python.

Quantum Operation Verification: Given the nature of the project, a significant focus will be on unit tests to verify different quantum operations. Unit tests will be written to confirm that individual quantum gates and operations produce the expected output.

Circuit Simulation Tests: The core functionality of the quantum circuit simulator involves simulating quantum circuits. The test suite will include comprehensive tests for circuit simulations, covering scenarios with various gate configurations, qubit states, and circuit depths to guarantee accurate and reliable simulation results.

Chapter 4

Design

This chapter will discuss the design of the application, and justify the decisions made in the design process.

4.1 Back-end

Django was planned to be primarily responsible for the design of the back-end of the application. To keep the focus on the main requirements of the application, where possible, the default Django settings and configurations would be used. The back-end was designed to be as simple as possible, with the primary focus on the simulation engine and the communication between the front-end and back-end.

4.1.1 Database Design

A small and simple database was required for the handling of user accounts and the storage of user-made quantum circuits. The database was planned to be designed using Django's built-in ORM (Object-Relational Mapping) system and would consist of two primary tables: the *User* table and the *File* table. The *User* table would store a user's first name, last name, email address and password. The *File* table would store circuit files, which would be stored as JSON objects, a title, a description, and a foreign key to the user who created the file. The ERD (Entity-Relationship Diagram) is shown in Figure 4.1.

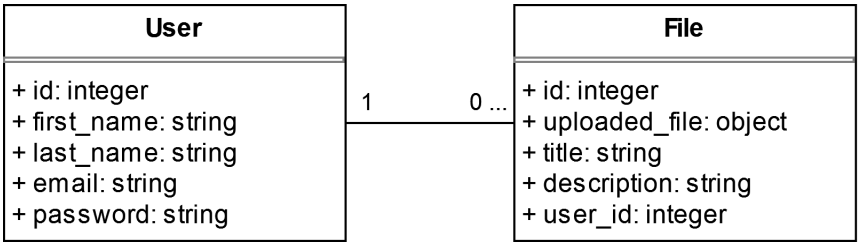


Figure 4.1: Database ERD diagram.

4.1.2 Simulator Design

The simulator was designed to be a class in Python, that would be responsible for the simulation of quantum circuits. It would take a request from the front-end of the application, containing circuit information, translate this into a Cirq circuit, and then simulate the circuit. The simulator would then return the results of the simulation to the front-end. The basic structure of the simulator class is shown below:

- Attributes:
  - `circuit`: The circuit to be simulated.
  - `qubits`: The number of qubits in the system.
  - `simulator`: The Cirq simulator to be used.
- Methods:
  - `load_circuit()`: Loads the circuit as a Cirq circuit.
  - `simulate_circuit()`: Simulates the circuit and returns the results.

4.2 Front-end

4.2.1 Design and Layout

The design and layout of the application were carefully considered to ensure that the application is user-friendly and accessible. While a mobile-first approach to design is typically recommended, the application was designed with a desktop-first approach. This is because a desktop-first approach would honor the foundational aspect of the application, which is the circuit builder. The circuit builder is the most complex feature of the application, and a desktop layout would provide the user with the most space and functionality to build and simulate quantum circuits.

Desktop Wireframes

Wireframes for the layout and basic design of the desktop application are shown in Figures 4.2, 4.3, 4.4, and 4.5.

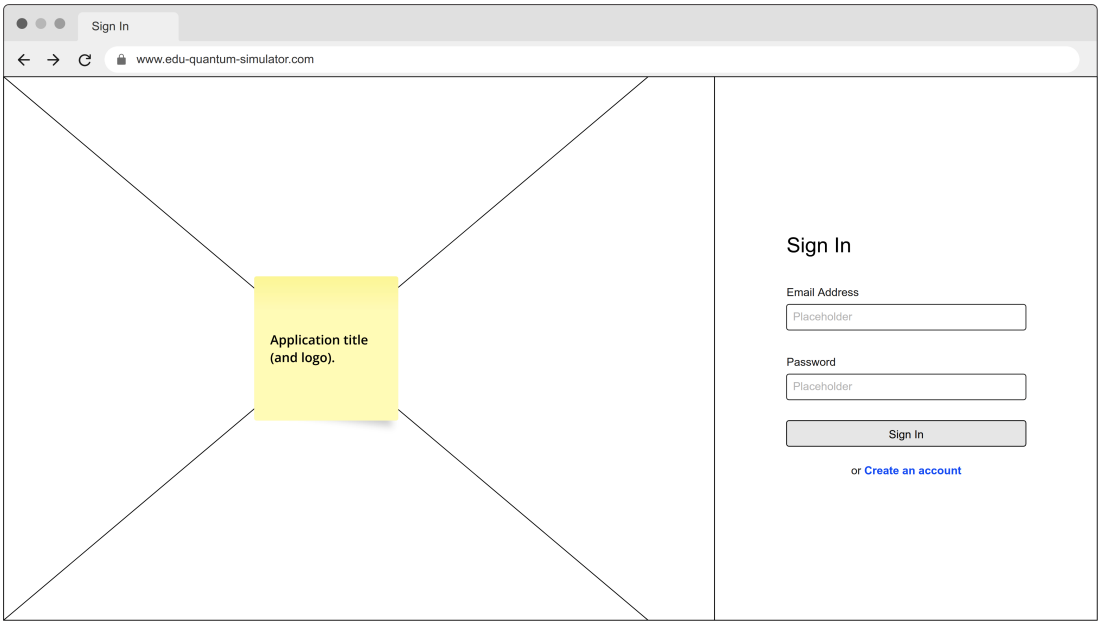


Figure 4.2: Desktop login page wireframe.

The wireframe in Figure 4.2 shows the layout of the login page, which was designed purposefully to be minimal. User accounts should be basic and require little information to create, and the login design reflects this.

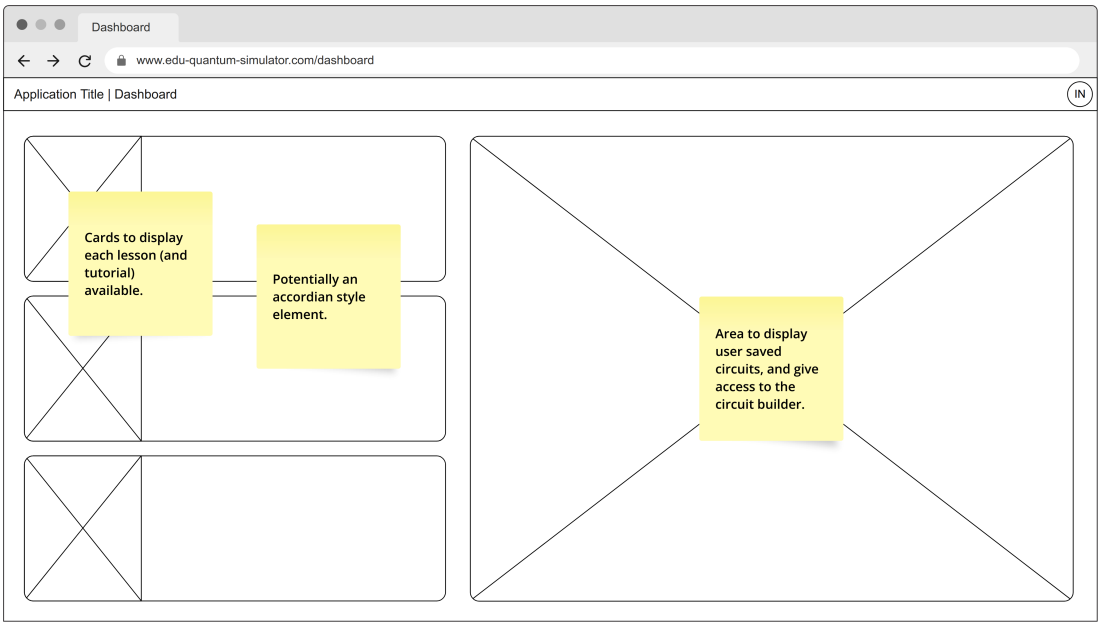


Figure 4.3: Desktop dashboard page wireframe.

Following the same principle as the login page, the dashboard page wireframe in Figure 4.3 was designed to be minimal and simple. The dashboard should provide a clear overview of the application's features, and advertise the educational content available to the user.

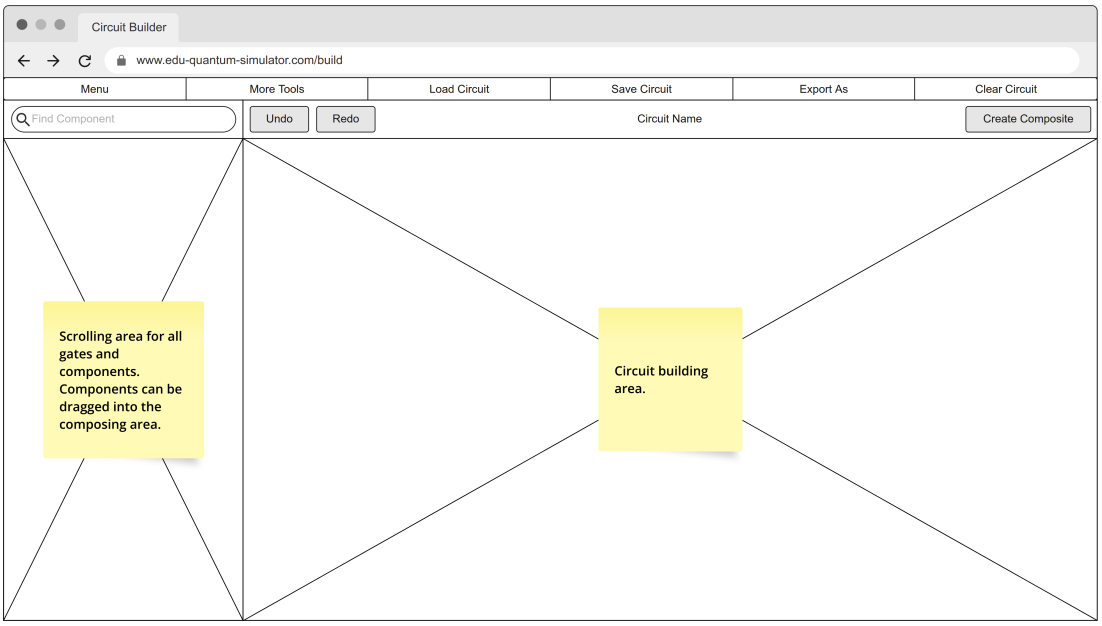


Figure 4.4: Desktop circuit builder page wireframe.

The circuit builder is the main feature of the application and the most complex page. The wireframe in Figure 4.4 shows the layout of the page, which has, again, purposefully been kept simple to avoid overwhelming the user. The primary aim of this design is to provide a clear and intuitive interface tool, while also providing the user with the necessary functionality to build and simulate quantum circuits. The use of collapsable sections, hidden panels or similar features would be avoided to help maintain the simplicity of the design.

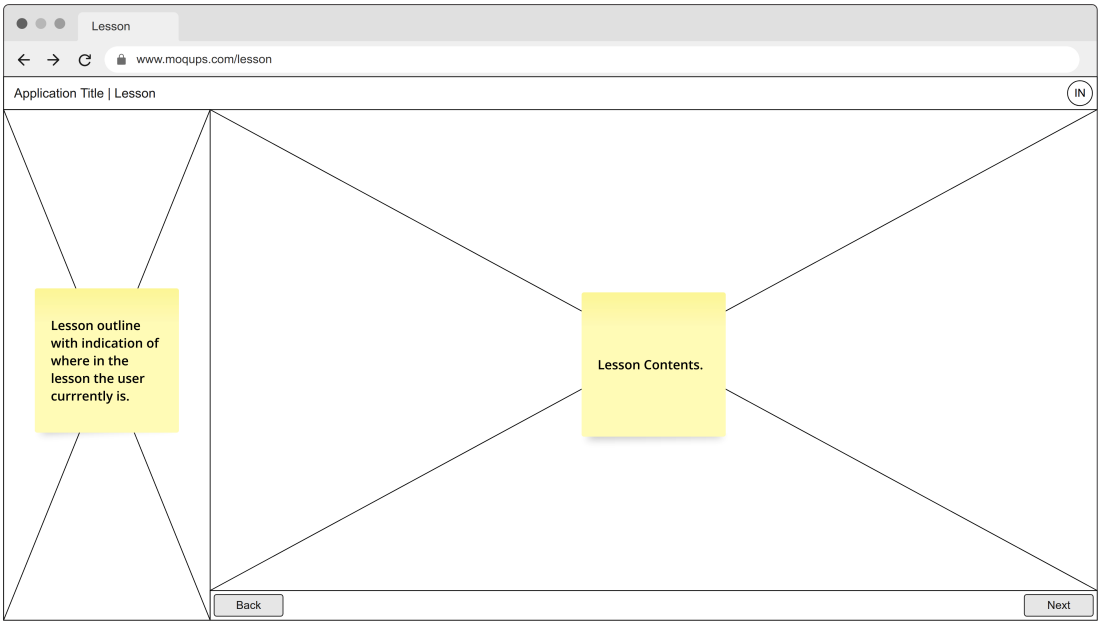


Figure 4.5: Desktop lesson page wireframe.

The lesson page as shown in Figure 4.5 was planned to provide the user with a clear outline of the lesson content. It would be built as a 'book-style' interface, where the user can navigate through small sections of content at a time.

Mobile Wireframes

Wireframes for the layout and basic design of the mobile application are shown in Figures 4.6, 4.8, 4.9, 4.7, and 4.10.

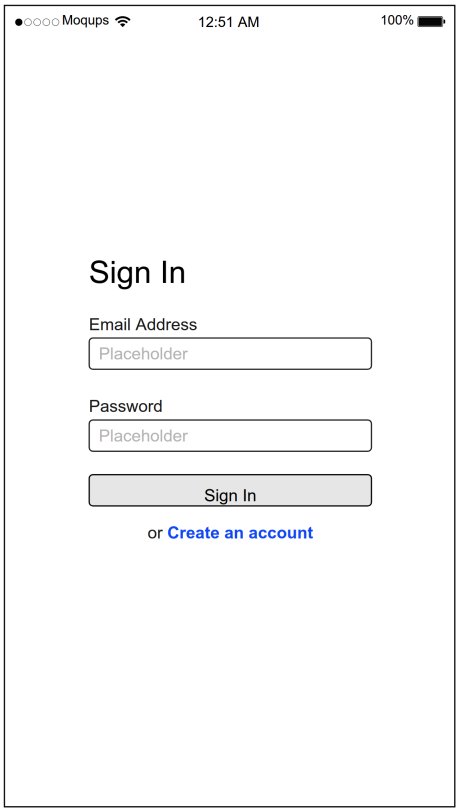


Figure 4.6: Mobile login page wireframe.

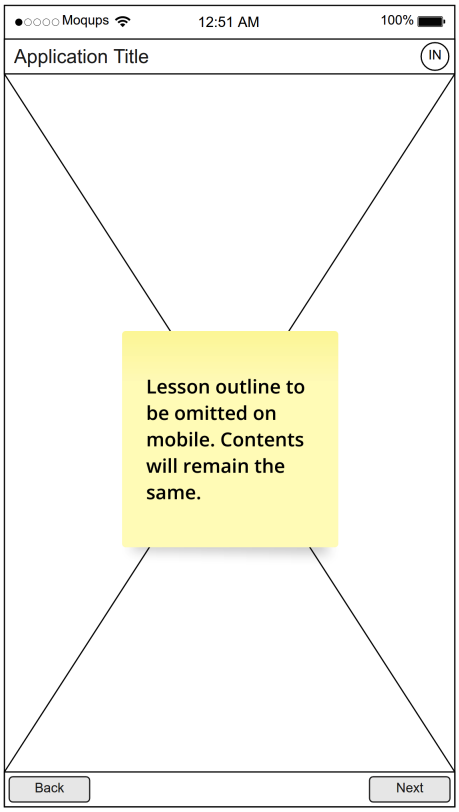


Figure 4.7: Mobile lesson page wireframe.

The login page and lesson page wireframes for the mobile application are shown in Figures 4.6 and 4.7 respectively. Following the same design principles as the desktop application, the mobile application was designed to be minimal and simple. The login page would provide the user with a clear and easy way to log in, and the lesson page would provide the user with clear and digestible content. The lesson outline would be omitted from the primary page view, to avoid cluttering the user interface, and would be accessible through a menu or similar feature.

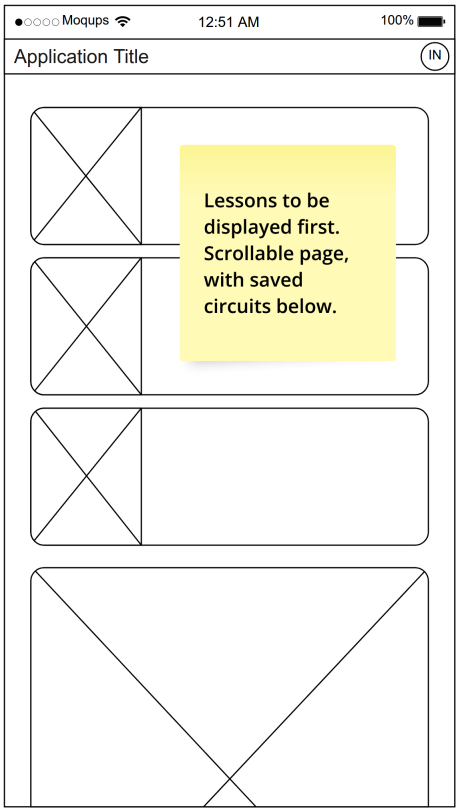


Figure 4.8: Mobile dashboard page wireframe.

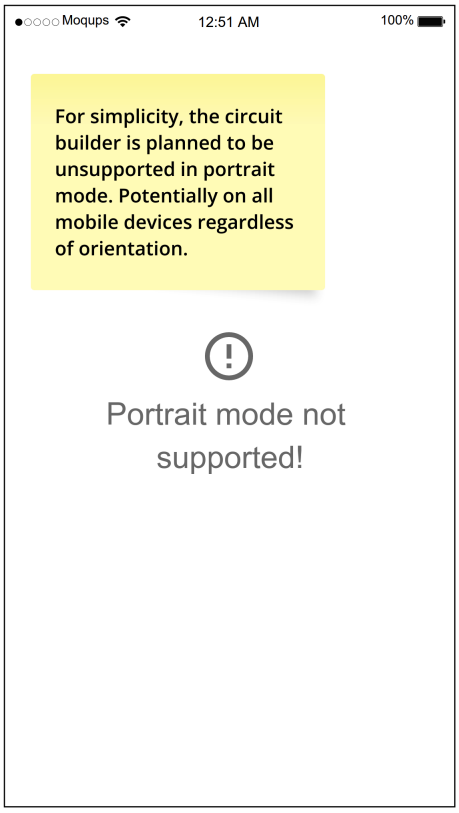


Figure 4.9: Mobile circuit builder page wireframe (portrait).

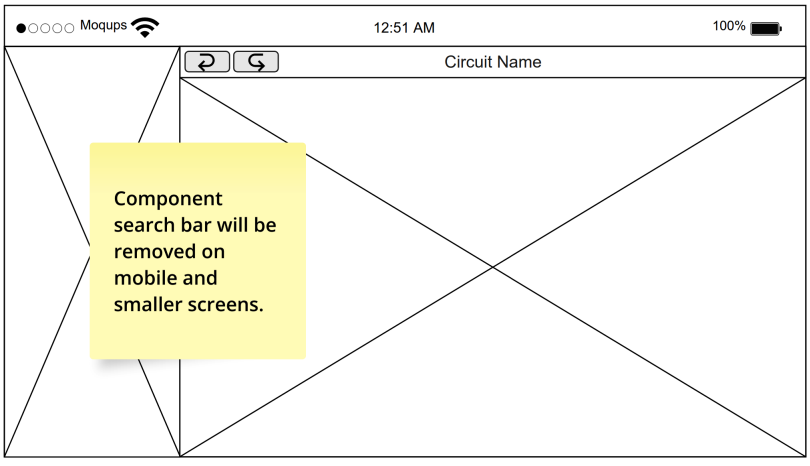


Figure 4.10: Mobile circuit builder page wireframe (landscape).

The dashboard and circuit builder wireframes for the mobile application are shown in

Figures 4.8, 4.9 and 4.10 respectively. The dashboard page would provide the user with a clear overview of the application's features, and the circuit builder would provide the user with the necessary functionality to build and simulate quantum circuits. The circuit builder was designed to prompt the user to rotate their device if they are using a mobile device in portrait mode, as shown in Figure 4.9. This would provide the user with a better experience and more space to build and simulate quantum circuits. If time had permitted, touchscreen functionality was planned to be implemented to allow the user to interact with the circuit builder on mobile devices when in landscape orientation.

4.2.2 User Flow

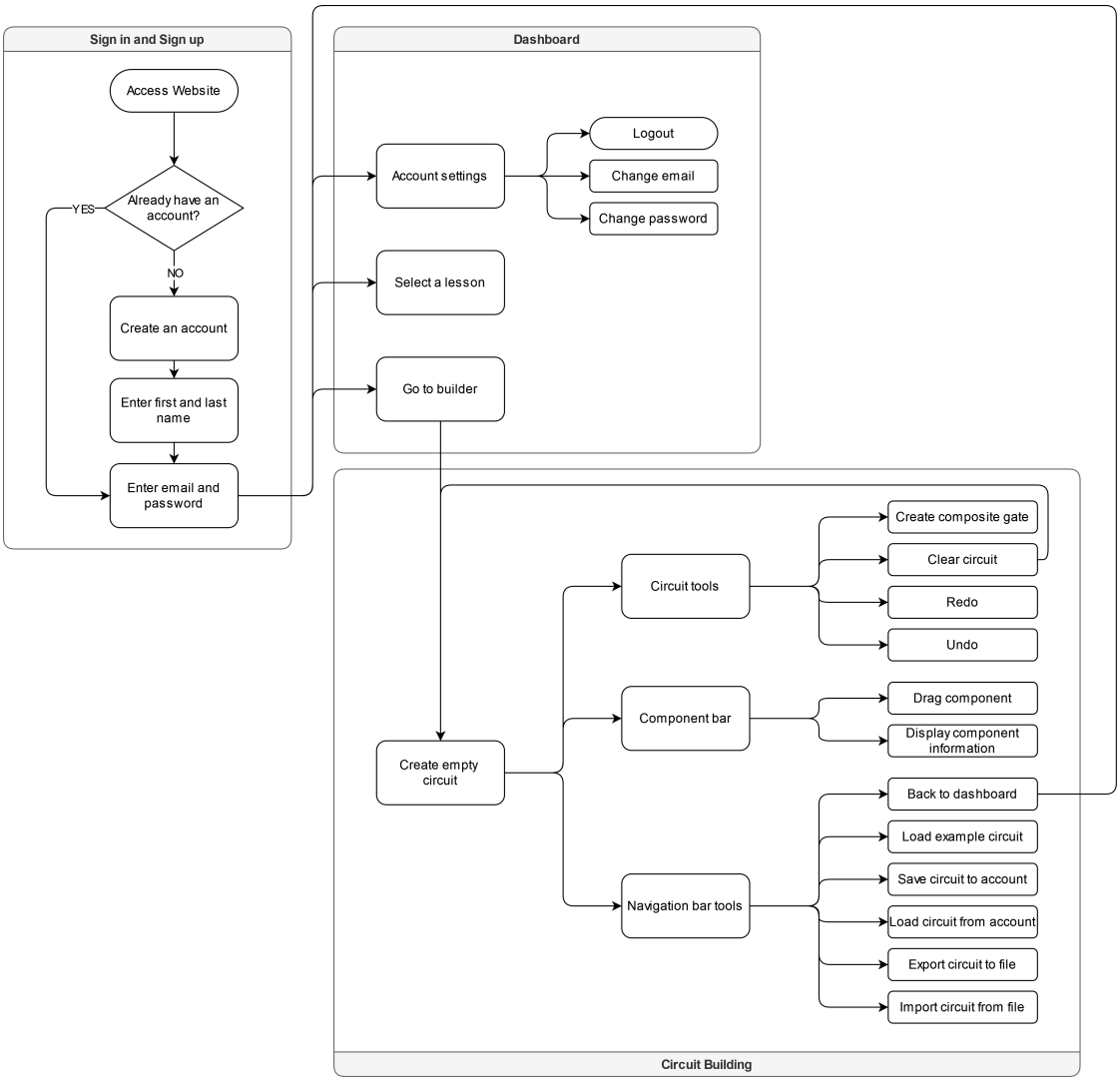


Figure 4.11: User-flow diagram.

The user flow of the application shown in Figure 4.11 expresses the primary user journey through the different application pages. The user would start at the login page, where they would be prompted to log in or create an account. Once logged in, the user would be directed to the dashboard, where they would be able to access the circuit builder, lessons, and their saved circuits. The user would be able to navigate between the dashboard, circuit builder, and lessons pages freely, and access their saved circuits from both the dashboard and the circuit builder.

Chapter 5

Implementation and Testing

This chapter will detail the implementation of the project, including the final technology stack, back-end and front-end implementation, deployment, and testing. Major decisions made during the implementation will be discussed, and the primary issues faced during development will be highlighted.

5.1 Implementation

5.1.1 Final Technology Stack

The final technology stack for the project is shown in Figure 5.1. The back-end was implemented using Django as planned, however, the original design to use a framework for the front-end (Vue.js) was changed, and no front-end framework was used. This decision was made to simplify the project and ultimately made the application more lightweight and easier to maintain. Instead, Tailwind CSS [38] was used to style the application, and vanilla JavaScript implemented much of the circuit-building logic.

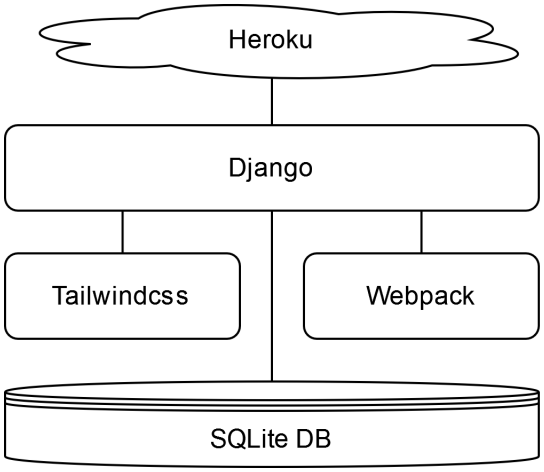


Figure 5.1: Final technology stack.

The additional use of Webpack [42] was also chosen during development, as it allowed multiple JavaScript modules to be bundled together and served by Django accordingly. This reduced the number of JavaScript files that needed to be served by Django.

5.2 Back-end

5.2.1 Overview

The back-end was implemented using Django, as planned, and the simulation engine was implemented using Cirq. A large proportion of the back-end was handled by Django's built-in features, such as the ORM (Object-Relational Mapping) system, which was used to design the database, and the views and models that were used to handle the application's logic.

5.2.2 Simulation Engine

A `Simulator` class was implemented to handle the simulation of quantum circuits. The class was responsible for generating a Cirq circuit from the circuit data, simulating the circuit, and generating measurement values for any measurement probes placed in a circuit. The basic structure of this class is shown below:

- Attributes:
  - `gateMap`: A mapping of the gates to their Cirq counterparts.
  - `rowNum`: The number of rows in the circuit.
  - `gates`: The circuit data.
  - `to_measure`: The user-selected qubits to measure.
  - `cirq_circuit`: The Cirq circuit.

  - `probed_circuit`: An additional Cirq circuit for probing measurements.
  - `qubits`: List of qubits in the system.
  - `simulator`: The Cirq simulator.

- Methods:
  - `generate_cirq_circuit()`: Generates the Cirq circuit from the circuit data.
  - `generate_probed_circuit()`: Generates the probed Cirq circuit.
  - `__parse_column()`: Parses a column of gates and adds them to the Cirq circuit.
  - `__find_controls()`: Finds the indices of control and anti-control gates in a given column.
  - `__find_swaps()`: Finds the indices of swap gates in a given column.
  - `simulate_circuit()`: Simulates the circuit and returns the state vector.
  - `__measurement_indices()`: Returns the indices of the qubits to measure.
  - `probe_measurements()`: Probes the circuit at the measurement points.
  - `__transpose()`: Transposes a given 2D array.

The simulation logic on the back-end was more complex than originally planned for, and the estimated effort for this task was underestimated. The implementation of the simulation engine was one of the primary reasons why some of the additional features were not implemented, such as the '*Create Composite Gate*' button. The decision to focus on the core functionality of the application was made to ensure the application was as complete as possible, and that the simulation engine was correct and functioning as expected.

5.2.3 Communication with the Front-end

The back-end communicates with the front-end using a dedicated view function — the `simulate` function, shown in Listing 1. It was responsible for receiving a POST request from the front-end, containing the circuit data and the qubits to measure. The circuit data was then passed to an instance of the `Simulator` class, which generated the Cirq circuit and simulated it. The results were then returned to the front-end as a JSON response.

```
@csrf_exempt
def simulate(request):
    try:
        if request.method == 'POST':
            data = json.loads(request.body)

            circuit_obj = data.get('circuit', [])
            circuit_array = circuit_obj.get('gates', [])
            to_measure = data.get('to_measure', [])

            simulator = Simulator(circuit_array, to_measure)
            simulator.generate_cirq_circuit()
            state_vector = simulator.simulate_circuit()

            simulator.generate_probed_circuit()
            probed_values = simulator.probe_measurements()

            return JsonResponse({
                'status': 'ok',
                'state_vector': state_vector.tolist(),
                'probed_values': probed_values })

    except Exception as e:
        return JsonResponse({
            'status': 'error',
            'message': str(e) })

    return JsonResponse({
        'status': 'error',
        'message': 'Invalid request method' })
```

Listing 1: The `simulate` view function.

5.2.4 The Order of Implementation

The back-end was implemented entirely after the front-end, which was an adequate approach, however, it would have been beneficial to implement parts of the back-end first. The back-end therefore had to be implemented **around** the front-end, for example, it needs to transform and parse the circuit data from the front-end more than what should have been necessary.

The `__transpose()` function is an artifact of this oversight and could have easily been omitted if the front-end had been developed with a better understanding of the back-end —

simply developing the front-end to store the circuit data column-wise rather than row-wise would have removed the need for this function. Another example of this is the handling of probability probes[1] within the circuit data which required verbose and complex logic to evaluate each probe. Two Cirq circuits are generated if probability probes are present, one for the simulation and one for calculating probe probabilities, which was not the most efficient solution. Other cases of this can be found throughout the back-end.

Side-by-side development, or more robust planning of the back-end structure, would have been beneficial to the project — deeper research into the Cirq library would have been a low-cost and high-reward solution, allowing the front-end to still have been developed first, but with a better understanding of how the back-end would be implemented.

5.3 Front-end

5.3.1 Overview

As mentioned in Section 5.1.1, the choice to use vanilla JavaScript over a front-end framework was made, Tailwind CSS was used for styling, and Webpack was used to bundle the JavaScript modules. The outline of the application's JavaScript modules is shown below:

- `circuit.bundle.js` — The circuit JavaScript bundle, serves only on the circuit builder page and contains the majority of the application's JavaScript code.
  - `circuit_area.mjs` — The primary module for the circuit builder, containing the construction of the circuit area, components, and functions to clear and update the circuit.
  - `circuit.mjs` — The module for the circuit, containing the `Circuit` class, with functions to add and remove gates from the circuit.
  - `gate.mjs` — The module for dragging (moving) gates, containing the `Gate` class, with functions to follow the cursor, snap to the circuit, and to drop.
  - `import_export.mjs` — The module for importing and exporting circuits.
  - `load_save.mjs` — The module for loading and saving circuits to a user account.
  - `talk_to_cirq.mjs` — The module for sending requests to the back-end to simulate the circuit.
  - `tutorial.mjs` — The module for the spotlight tutorial.
  - `graphs.mjs` — The module for displaying the probability histogram (that uses ApexCharts [1])
  - `hover_info.mjs` — The module for displaying corresponding information in the '*Information*' area when hovering over components.

[1] Where, for clarification, a probability probe is a circuit component that will display the probability of a qubit being in the $|1\rangle$ state at the position of the probe. In this specific case, a probe is purely for inspection and learning purposes and does not perform a measurement that collapses the qubit.

  - `icons.mjs` — The module for generating SVG icons that are used throughout the application.

- `lesson.bundle.js` — The dashboard JavaScript bundle, served on all the lesson pages.
  - `lesson.mjs` — The module for the lessons, containing the lesson class, with functions to navigate through the lesson content and manage quizzes.

- `main.bundle.js` — The general JavaScript bundle, served on all pages of the application.
  - `ajax_tables.mjs` — The module for managing the saved circuit tables.

5.3.2 A Canvas Approach to the Circuit Builder

The initial approach for the circuit builder GUI was to use the JavaScript Canvas API [25], where the circuit would be drawn onto a canvas element. This approach was abandoned for many reasons, primarily due to the complexity of the implementation and the difficulty in managing the circuit components. Secondly, the Canvas approach required a considerable effort to style appropriately and did not scale well with additional features. This approach cost the project substancial time and delayed the implementation by approximately 2 weeks.

5.3.3 Communication with the Back-end

The communication was implemented using JavaScript's Fetch API [26], which allowed the front-end to send requests to the back-end and receive responses. A single and intuitive function was implemented to send requests to the back-end to simulate the quantum circuit. The `simulate` function in Listing 2 sends a POST request to the back-end, containing the circuit data and the qubits to measure. The response is then parsed and stored in the front-end's state, ready to be displayed to the user.

```
export async function simulate(circuitData, toMeasure) {
    try {
        const response = await fetch('/simulate', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({
                circuit: circuitData,
                to_measure: toMeasure,
            }),
        });
        const data = await response.json();
        currentStatus = data.status;
        currentStateVector = data.state_vector;
        currentMeasurementProbeValues = data.probed_values;
    } catch (error) {
        currentStatus = 'error';
        console.error('Error:', error);
    }
}
```

Listing 2: The `simulate` function.

5.3.4 Final Design and Layout

The final page design and layout closely followed the wireframes, with some minor changes to improve the user experience. The login and sign-up pages shown in Figures A.1 and A.2 depict this. A minor change seen in the login page was the addition of a '*Continue as Guest*' button, which allows users to access the application without creating an account. This decision was made to make the application more accessible to users who may not want to create an account and to allow users to explore the application before committing to creating an account. This choice was also partly influenced by the user feedback survey — allowing users as easy access as possible to the application was a priority, to maximise the number of participants in the survey.

5.3.5 Dashboard

Logging in, or continuing as a guest, will direct the user to the dashboard page, shown in Figures A.3 and A.4. The dashboard has slight differences between the guest and logged-in versions, with the logged-in version showing the user's saved circuits and lessons. The guest version does not have this feature and instead prompts the user to log in or create an account to access these features. Access to the circuit builder and lessons is not restricted for guests, and they can access these features freely as seen in Figures A.3 and A.4.

5.3.6 Circuit Builder

The circuit builder page, shown in Figures A.5 and 5.2, allows the user to build and simulate quantum circuits. It also closely follows the wireframes, with some changes. Most notably, the originally planned '*Undo*', '*Redo*' and '*Create Composite Gate*' buttons are removed. Both '*Undo*' and '*Redo*' were never explicit requirements, and the nature of the drag-and-drop functionality made removing and replacing components simple and easy without the need for these buttons. Components can be dragged onto the circuit, and removed by either clicking them or dragging them away from the circuit.

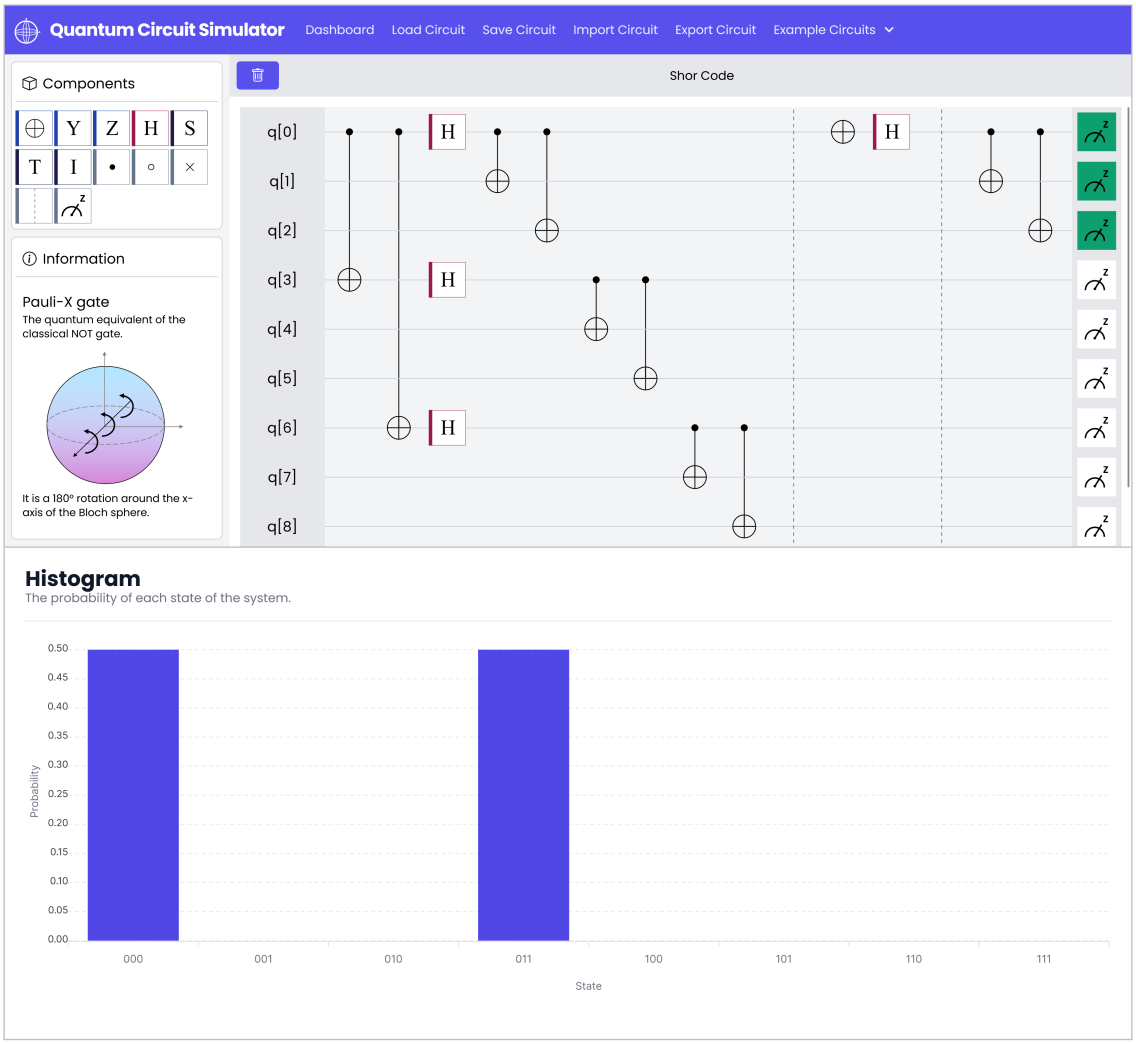


Figure 5.2: Desktop circuit builder (partially) displaying Shor Code.

The '*Create Composite Gate*' button was removed due to time constraints (mentioned previously in Section 5.2.2), and the complexity of implementing this feature. The decision was made to focus on the core functionality of the circuit builder, in which being able to

create a composite gate does not affect the completeness of the simulator.

The circuit builder features useful tooltips when hovering over components and buttons. When hovering over a component, a dedicated '*Information*' area will be populated with helpful visualisations (where applicable) and a description as seen in Figures 5.3 and 5.4.

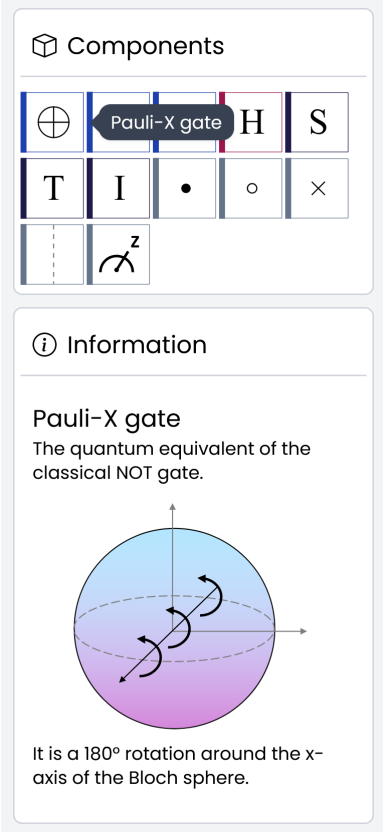


Figure 5.3: Pauli-X gate example of the circuit builder information section.

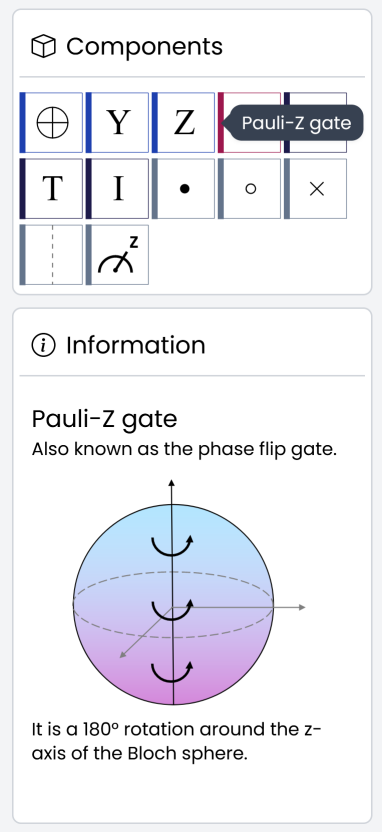


Figure 5.4: Pauli-Z gate example of the circuit builder information section.

5.3.7 Spotlight Tutorial

A spotlight-style tutorial was implemented to guide the user through the circuit builder and explain the different components and features. An example of the tutorial is shown in Figure A.6, and it is accessible from the dashboard page.

5.3.8 Lessons

A series of 3 lessons were created, designed to be taken in order, to provide the user with a basic understanding of quantum computing:

1. 'Introduction to Quantum Computing' — An introduction to quantum computing, that builds on a knowledge of classical computing.

2. 'Fundamentals of Quantum Computing' — Looking further into single and multi-qubit gates, and how they can be used to build quantum circuits.

3. 'Quantum Circuits' — Constructing quantum circuits and how gates can be combined.

Examples of the lessons are shown in Figures A.7 and A.8. Each lesson features an overview of the content, and a 'book-style' navigation system, allowing the user to navigate through the content in small sections at a time. The lessons also feature interactive quizzes, to test the user's knowledge. An example of a quiz is shown in Figure A.9.

5.3.9 Mobile Layout

The final mobile layout similarly follows the wireframes closely and can be seen in Appendix B. The refinement and perfection of the mobile design was limited due to time constraints, and the focus was on ensuring the application was functional and accessible on mobile devices. The circuit builder layout, in particular, was not optimised for mobile devices (Figures B.5 and B.6). It is simply the desktop design scaled down to fit the smaller screen size, which appears cluttered and difficult to use. As discussed in Section 4.2.1, a desktop-first approach was decided upon during the design phase, and the final mobile layout reflects this decision.

5.4 Deployment

The application was deployed using Heroku [19], chosen for its simplicity and ease of use. Deployment was basic, using a single web dyno[2]. The decision to deploy was made during development, and more specifically, prioritised over R14 (containerise the web application into a desktop version — see Table 3.5). The choice of deployment over implementing R14 aided in the collection of user feedback, as it allowed the application to be accessible remotely to users, greatly widening the potential participant pool for the user feedback survey. This was particularly important as the survey would be one of the key indicators of the application's success. Additionally, due to the time constraints of the project, deployment was a more feasible option than containerising the application into a desktop version.

5.5 Testing

A variety of testing methods were used to ensure the application was both functioning correctly and met the project requirements. These methods included unit testing, functional testing, user acceptance testing, and a Lighthouse audit.

5.5.1 Unit Testing

Unit testing was important, primarily to ensure aspects of the application concerned with quantum simulation and calculations were valid and correct as mentioned in Section 3.3.1.

[2] An isolated Linux container.

The back-end was tested using Django's built-in testing framework, and the front-end was tested using Jest [21] and the Jest-DOM [39] library. 100.00% coverage was achieved for the back-end, as shown in Figure 5.5.

Module ↑	statements	missing	excluded	branches	partial	coverage
confighome\simulator.py	103	0	0	68	0	100%
confighome\views.py	138	0	0	44	0	100%
users\admin.py	15	0	0	0	0	100%
users\apps.py	4	0	0	0	0	100%
users\forms.py	30	0	0	2	0	100%
users\managers.py	20	0	0	6	0	100%
users\models.py	18	0	0	0	0	100%
Total	**328**	**0**	**0**	**120**	**0**	**100%**

Figure 5.5: Unit testing coverage report for the application's back-end.

Coverage for the front-end was not as high, with 89.32% coverage achieved, as shown in Figure 5.6. This was primarily due to the time constraint of the project, and the focus for the front-end testing was placed on functional files, rather than those dedicated to the generation of HTML elements and styling. The `circuit_area.mjs` module has been omitted from the coverage report for this reason.

File	Statements		Branches		Functions		Lines	
talk_to_cirq.mjs	100%	11/11	100%	0/0	100%	1/1	100%	11/11
load_save.mjs	100%	19/19	100%	4/4	100%	3/3	100%	19/19
icons.mjs	100%	219/219	100%	16/16	100%	16/16	100%	217/217
circuit.mjs	100%	144/144	100%	56/56	100%	18/18	100%	118/118
tutorial.mjs	99.03%	103/104	86.66%	13/15	100%	8/8	100%	101/101
import_export.mjs	95.65%	132/138	75%	18/24	86.66%	13/15	95.41%	125/131
hover_info.mjs	95.12%	39/41	50%	1/2	86.66%	13/15	94.87%	37/39
graphs.mjs	92.3%	24/26	100%	4/4	71.42%	5/7	91.3%	21/23
gate.mjs	86.15%	56/65	66.66%	8/12	66.66%	8/12	88.33%	53/60
ajax_tables.mjs	80%	12/15	100%	2/2	87.5%	7/8	80%	12/15

Figure 5.6: Unit testing coverage report for the application's front-end.

5.5.2 Functional Testing

Functional testing was conducted manually across a range of operating systems and browsers, to ensure the application was functioning as expected. Testing for backward browser

compatibility was not conducted, as this was deemed out of the scope of the project. The application was tested on:

- Windows 11
  - Firefox (125.0.2).
  - Chrome (124.0.6367.92).
  - Edge (124.0.2478.67).
- iPadOS 17.4.1
  - Safari.
  - Chrome (124.0.6367.88).
- iOS 16.3.1
  - Safari.
  - Chrome (124.0.6367.88).

Most notably, the circuit builder is not compatible with touchscreen devices, regardless of whether it is on a desktop or mobile device. This was explicitly intentional in the design for mobile devices, however, this is a feature that should be implemented in the future for desktop devices to improve the accessibility of the application. Overall the application functioned as expected across all devices and browsers, with no major issues found.

5.5.3 User Acceptance Testing

User acceptance testing was loosely conducted as part of a user feedback survey (Appendices C, D, E and F). The survey was completed by 17 participants, who were asked to complete a series of tasks on the application and provide feedback on their experience. In-depth analysis and discussion of the survey results can be found in Section 6.1. Table 5.1 shows the survey questions used for the user acceptance testing, and the responses given by the participants. The final row of the table is concerned with the performance of the application and is therefore not directly correlated to a requirement. Overall, the responses were acceptable and each test case was passed (a score of 4+) by a majority of participants.

Requirement ID	Description	Survey Question	Average Response
R1, R13	Upon completion of the interactive lessons, the user should be more informed of quantum concepts.	I understand more about quantum computing concepts now than I did prior to this experience.	5.00
R6	Upon exploration of the circuit builder, the user should find the additional information helpful and informative.	The circuit visualisation is easy to understand.	4.00
		The probability visualisations are easy to understand.	4.29
n/a	From start to end of accessing and leaving the application, the user should find the load time(s), and general usage to be at an acceptable speed.	The loading times for the platform are reasonable.	4.71

Table 5.1: Table showing the explicit survey questions used for loose user acceptance testing.

5.5.4 Lighthouse Audit

An audit was conducted using Google's Lighthouse tool [18] to assess the performance, accessibility and best practices of the application. Two of the five possible metrics, SEO (Search Engine Optimisation) and PWA (Progressive Web App) were not included in the audit as they are not within the scope of the project, and do not serve any indication of success regarding the project's aims or requirements. The audit was conducted on the deployed application, over all pages of the platform, and the results are shown in Table 5.2.

Page	Desktop			Mobile		
	Performance	Accessibility	Best Practices	Performance	Accessibility	Best Practices
/login	91	100	100	57	100	100
/signup	91	100	100	66	100	100
/builder	88	100	100	n/a*	100	96
/dashboard	94	100	100	40	100	100
/reset/done	91	100	100	n/a*	100	100
/password_reset/done	85	100	100	n/a*	100	100
/reset/MQ/set-password	88	100	100	n/a*	100	100
/password_reset/	88	100	100	n/a*	91	100
/<all lessons averaged>	77	100	100	42	100	100
/change_email/	96	100	100	58	100	100
/change_email/done	95	100	100	60	100	100
/password_change/	98	100	100	56	100	100
/password_change/done	71	100	100	71	100	100
Average	89	100	100	35	99	100

Table 5.2: Lighthouse audit score summary.

On desktop, the application performed well in both accessibility and best practices, scoring 100% in both categories. The performance score was slightly lower, at 89%, however, this is still acceptable and does not indicate any major issues with the application's performance. On mobile, the accessibility and best practices scores were 99% and 100% respectively. The performance score, however, was significantly lower, at 35%[3]. One of the contributing factors to the performance scores on both desktop and mobile was the deployment of the application. The application was deployed on minimal resources (as Heroku is a paid service), and the performance could be improved by deploying on a more powerful server with more dedicated resources. Despite this, mobile is particularly low and has many areas for improvement. While mobile performance was out of scope for this project, its development would improve the application's accessibility to all users and should be considered for future development.

Overall, the audit results are positive, with the application scoring highly in accessibility and best practices which aligns well with the project's primary goal of creating an accessible and user-friendly application.

5.6 Summary of the Requirements Met

Table 5.3 shows a summary of the requirements met after implementation — 10 out of the 14 requirements were fully met. As mentioned in Section 5.3.6, requirements R9 and R10 were not implemented due to time constraints, and the complexity of the implementation. However, the creation of composite gates is a feature that would be beneficial to the user experience and should be considered for future development. Additionally, requirement R13 was not implemented similarly due to time constraints. As requirement R1 was already met, the implementation of R13 was not a priority during development and was ultimately omitted.

[3]*Where a n/a score indicated an error in the audit, counted as a 0.

Finally, requirement R14 was not implemented, as the decision to deploy the application was made over containerising the application into a desktop version (Section 5.4).

ID	Requirement	Met	Comment
R1	The application must contain a series of interactive lessons that teach the user about quantum computing fundamentals.	✓	There are a total of 3 interactive lessons available on the platform teaching the fundamentals of quantum computing.
R2	The application must be able to simulate two-qubit gates.	✓	Two-qubit gates can be composed by grouping nodes and/or one-qubit gates together on the application.
R3	The application must be able to simulate single-qubit gates.	✓	Pauli-X, Y and Z gates, S and T-gates, the Identity gate and Hadamard gate have been implemented.F9
R4	The application must allow multiple gates to be combined in parallel, series and or both.	✓	Circuits of to 9 qubits are permitted, allowing parallel and/or series connections to be made.
R5	The application must provide the user with a way to visualise qubit states.	✓	A final system state histogram is available, along with probability probe components.
R6	The application must provide the user with useful information about simple gates and the state of a selected qubit.	✓	Tool-tips for every component in the builder, and an additional information area have been implemented
R7	The application must contain a walkthrough tutorial to introduce the user to the environment and indicate how to use the application.	✓	A spotlight-style tutorial has been implemented.
R8	The application should be able to simulate multi-qubit gates.	✓	Multi-qubit gates can be composed by grouping nodes and/or one-qubit gates together on the application.
R9	The application should allow a collection of simple gates to be compiled into a composite gate.	✘	Not implemented due to time constraints.
R10	A user should be able to save and 're-use' a created composite gate.	✘	Not implemented due to time constraints.
R11	The application should provide the user with exemplar circuits of quantum algorithms.	✓	Examples of bit-flip, phase-flip, Shor code, a 3-qubit fourier transform, and entanglement examples are included.
R12	A user should be able to save a circuit to a file to allow circuits to be saved, shared, and modified.	✓	Users can download and upload JSON circuit files, or save and load circuits from their account.
R13	The application could contain an additional series of interative lessons that teach the user about more advanced concepts.	✘	Not implemented due to time constraints.
R14	The application could be containerised to be deployed as a desktop application as well as a website application.	✘	Not implemented due to time constraints.

Table 5.3: Summary of the requirements met after implementation.

Chapter 6

Results and Discussion

Results from the testing, alongside results from the user feedback survey, are analysed in depth in this chapter. The chapter will also discuss the goals achieved in the project, the limitations of this project, and suggest further work that could be done to improve the application.

6.1 User Feedback Survey Results

The user feedback survey was conducted to gather feedback from users on the application and serves as one of the primary indicators of the success of the project, particularly concerning its educational quality. As initially mentioned in Section 5.5.3, the survey was conducted with 17 participants, who were asked to access the application from a desktop device. The survey consisted of four main sections:

- Section 1 (required): Demographic information.
- Section 2 (required): Feedback on the circuit builder and simulator, given on a scale of 1 to 5 (1 being strongly disagree, 5 being strongly agree).
- Section 3 (optional): Feedback on the educational lessons, given on a scale of 1 to 5 (1 being strongly disagree, 5 being strongly agree).
- Section 4 (optional): Overall feedback on the application, given as free-text answers.

6.1.1 Demographic Information of Participants

The demographic of the participants is as follows:

- All participants were between the ages of 18 and 25.
- All participants were undergraduate students.
- 59% of participants said they study or work in a computer science-related field.

- All participants said they had either no prior knowledge or a basic understanding of quantum computing.
  - 82% of participants said they had no prior knowledge of quantum computing.
  - 18% of participants said they had a basic understanding of quantum computing.

The participants align very well with the target demographic of the application, which is undergraduate students, the majority studying a computer science field, with little to no prior knowledge of quantum computing.

6.1.2 Feedback on the Circuit Builder and Simulator

The responses to this section of the survey are visualised in Figure 6.1, where each group of bars along the x-axis represents the responses to a specific question, and the height of the bars represents the frequency of each response.

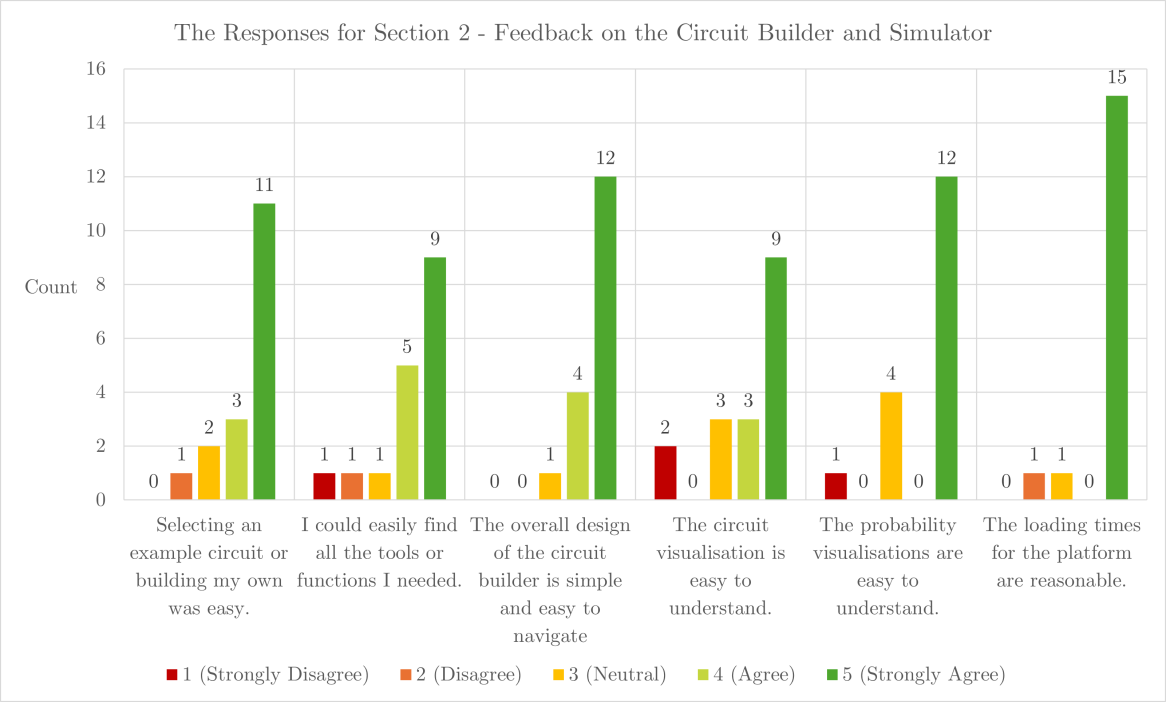


Figure 6.1: A bar graph to visualise the responses to Section 2 of the survey.

Keeping the demographic information in mind, the responses to the questions in this section are very positive. While the majority of participants had no prior knowledge of quantum computing, the responses show that the application was still easy to use and understand, with an average greater than 4.00 for all questions. Therefore, when we look at the response to the questions '*The circuit visualisation is easy to understand*' and '*The probability visualisations are easy to understand*', the average response of 4.00, and 4.29 respectively indicate strong

success in the original criteria of the application concerning visualisation and providing helpful information (requirements R5 and R6, see Section 3.2, Table 3.5).

Similarly, the responses to the questions '*Selecting an example circuit or building my own was easy*', '*I could easily find all the tools or functions I needed*' and '*The overall design of the circuit builder is simple and easy to navigate*' show that the application was easy to navigate and use, with an average response of 4.41, 4.18 and 4.65 respectively. This indicates that the application was successful in meeting the goals of being both user-friendly and accessible.

Finally, the response to the question '*The loading times for the platform are reasonable*' was exceptionally high, and had an average response of 4.71. This indicates that the user experience was not hindered by slow loading times (Section 3.3.1), and this is further supported by the results from the Google Lighthouse audit (Section 5.5.4).

6.1.3 Feedback on the Educational Lessons

14 out of the 17 participants chose to provide feedback on the educational lessons. The responses to this section of the survey are visualised in Figure 6.2, where the axes are the same as in Figure 6.1.

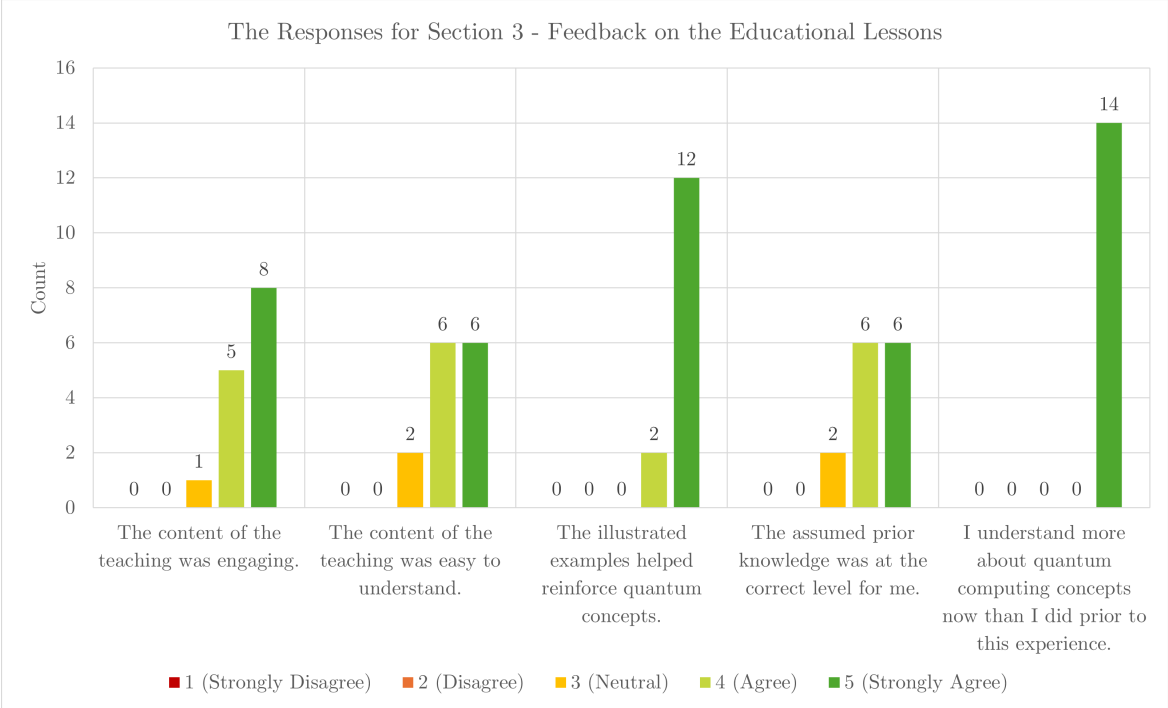


Figure 6.2: A bar graph to visualise the responses to Section 3 of the survey.

The responses to the questions in this section are also very positive, particularly in the final question, '*I understand more about quantum computing concepts now than I did prior to this experience*', where every participant strongly agreed (5.00 average). This consolidates

the overall success of the educational lessons in the application, where the primary goal was to provide an educational experience, to the same demographic of users as the survey participants, that would improve their understanding of quantum computing concepts (requirements R1 and R13, see Section 3.2, Table 3.5).

However, the responses to the questions '*The content of the teaching was easy to understand*' and '*The assumed prior knowledge was at the correct level for me*' both had the lowest average response of 4.29 for this section. While this is still positive, this indicates that while the educational content was successful in engaging the participants, some participants may have struggled to understand the content in its entirety. This suggests that there are subtle improvements that can be made to the educational content to improve its clarity and effectiveness.

6.1.4 Overall Feedback on the Application

Manual response coding was used to analyse the free-text answers given in Section 4 of the survey. The responses were used to code categories for the questions '*What did you like the most about the platform*?' and '*What improvements would you suggest for the platform?*'. The coding for the former question is shown in Table 6.1, and the derivation for these codes can be seen in Table G.1.

"What did you like the most about the platform?"		
id	Code	Count
c1	Easy to use	5
c2	Content was engaging	2
c3	Content was easy to understand	5
c4	Content was at the right level for me	1
c5	Design and layout is nice	7

Table 6.1: A table containing the categories chosen for coding the free-text question '*What did you like the most about the platform?*'

The most common code was **c5**, which occurred 7 times within the responses — showing strong support for the overall design and layout of the application. The codes **c2**, **c3**, and **c4** were also common, with 8 occurrences total between them. This indicates that the participants found the educational content and lessons to be informative, engaging and simple to understand. The code **c1**, with 5 occurrences, consolidates these findings, showing that the participants overall found the application easy to use and navigate.

Overall, the response to this free-text question supports the findings from the quantitative data, showing that the participants found the educational content to align with the goals of the application and that as a whole, the application was easy to use and navigate.

The coding for the final question, '*What improvements would you suggest for the*

platform?' is shown in Table 6.2, and the derivation for these codes can be seen in Table G.2.

"What improvements would you suggest for the platform?"		
id	Code	Count
c1	Incorrect information	1
c2	Quiz pop-ups look out of place	4
c3	Too much white space	3
c4	Something was hard to find	5

Table 6.2: A table containing the categories chosen for coding the free-text question '*What improvements would you suggest for the platform?*'

The most common code was **c4**, which occurred 5 times within the responses, showing that the biggest criticism participants had was that something was difficult to find. Specifically:

1. The navigation buttons for the lessons were not clear.
2. The probability histogram was not obvious in its placement.
3. The circuit builder tutorial was not easy to find.

The first point is related to the responsive layout of the application — there is only one breakpoint, so on larger screens, the navigation buttons are very small. This is one of the primary issues that would be prioritized in future work (discussed further in Section 6.5). The second point is related to the design of the application. The histogram is placed directly below the circuit builder, but this is not immediately obvious (Figure A.5). There is no indication that the histogram is there, and this is a simple design change that could be made in the future to improve the design of the application. The third point is similarly related to the design of the application, the tutorial is accessible from the user dashboard, but it blends in with the lessons (Figure A.4).

The code **c3** relates back to the responsive design of the application. Participants with large screens found that the application did not scale well, and it resulted in a large amount of white space. This is another issue that would be prioritized in future work (Section 6.5).

The code **c2** is for a specific design comment — the participants found the interactive quiz pop-ups to be ill-fitting with the rest of the application design. The decision was made during development to use default browser alerts for the quizzes to save time while retaining functionality. In the future, these should be redesigned using JavaScript and Tailwind CSS to fit the rest of the application design.

Finally, the code **c1** was included to address a particular comment on a quiz question. The participant who made this comment correctly found an error in a quiz question, and this should be rectified in the future.

6.1.5 Constraints of a User Survey

There are considerations to be made when conducting a user survey, most importantly the small sample size. The survey was conducted with 17 participants, which may not be representative of the entire target demographic of the application. Even though the demographic of the participants aligned well with the target demographic of the application, the small sample size means that the results may not be generalisable to the entire target demographic.

Additionally, the survey was conducted with participants who were asked to access the application from a desktop device. Although very positive feedback was received, it should be noted that this is specifically for the desktop experience of the application, and the mobile experience will be different.

Finally, there may be a recency bias within some participants. A few of the participants were recruited from the University of Sheffield, specifically others on the Computer Science course, completing their own Dissertation projects. This may have influenced the feedback received, as the participants may have been more likely to provide positive feedback to support a fellow student during this time.

In order to combat this bias several measures were taken. The survey was conducted anonymously, collecting as minimal demographic information as possible. Additionally, the section of the survey concerned with the educational content was intentionally optional, as were the free-text responses. These measures were intended to reduce the pressure on participants to provide positive feedback.

With these constraints in mind, it is important to remember that the survey does not reflect on the mobile experience of the application. However, the positive feedback received in the user survey is still strongly supported by the other forms of testing completed and is not to be discredited.

6.2 Challenges Faced

The primary challenge faced during the development of the application was time constraint. During development, many trade-offs were made to ensure that the project was completed on time, and this is reflected in the unmet requirements (Section 5.6). Additionally, the negative effect of the time constraint was increased by the decision made early on to approach the front-end using the JavaScript Canvas API (Section 5.3.2). This was abandoned in favour of using the Tailwind CSS framework, which was more efficient and user-friendly, but this initial decision resulted in a loss of development time.

Another factor that put strain on the project's development time-wise was the order in which the application was developed, and the lack of planning to support this. As mentioned in Section 5.2.4, the front-end of the circuit builder was developed before the back-end, which resulted in the back-end code being more complex than it needed to be — it had to be designed around the front-end, rather than with. This, again, resulted in a loss of development time.

Time constraints also affected the scope of the project. From the beginning, it was planned

to use the Cirq library for quantum computations, rather than implementing a standalone quantum simulator. This was a significant decision that was made to ensure that the project was completed on time, and while it was successful in this regard, it did limit the scope of the project. Further research should be done to determine the feasibility of implementing a standalone quantum simulator in the future, and if this would have been advantageous.

6.3 Goals Achieved

Despite the challenges faced throughout this project, many of the original goals and requirements were met. The original goals of the project (Section 1.1) are summarised as follows:

1. Develop an educational quantum circuit builder and simulator.
2. That is intuitive and user-friendly.
3. And provides guided resources aimed at educating the user on quantum computing and programming.
4. The platform will specifically target those with undergraduate-level knowledge.

The first, and most fundamental goal of the project was achieved and deployed successfully. The unit testing (Section 5.5.1) supports this, showing that overall the application is robust and reliable. Additionally, the validation and verification of the quantum operations of the platform was one of the primary concerns of the project (Section 3.3.1). The extensive unit testing of the simulator confirms that the application is correct and accurate.

The second goal was also achieved, with the extent of its success demonstrated by the user feedback survey (Section 6.1), specifically the feedback on the circuit builder and simulator. The response to the questions in this section of the survey shows that the application was easy to use and navigate, and the design was simple and intuitive, with an average response of 4.37 (out of 5) for all questions. The accessibility metric of the Google Lighthouse audit (Section 5.5.4) also supports this, with the application scoring an average of 100% on desktop, and 99% on mobile.

As outlined in Section 5.6, guided resources were implemented as a series of interactive lessons. However, they all focused on introductory and foundational information — due to time constraints more advanced lessons did not get implemented. Despite this, the feedback received in Section 3 of the user survey (Section 6.1.3) was very positive and emphasises the success of the third original goal of the project. Every participant who completed this section of the survey strongly agreed that they understood more about quantum computing concepts after surveying than they did prior.

The demographic statistics of the user survey participants (Section 6.1.1), align very similarly to that of the application's target audience. Given this and the very positive feedback, particularly in Section 3 of the survey indicates success in the final goal of this project. For example, an average response of 4.29 was received for the question '*The assumed*

prior knowledge was at the correct level for me', which demonstrates that for the application's target audience, the project has been particularly successful. Additionally, the free-text responses collected (Section 6.1.4) further support this, with applicants frequently writing that the application is 'easy to use', and that the 'design and layout is nice'.

6.4 Limitations

One of the primary limitations of this project is the current responsive design of the application. As mentioned in Section 6.1.4, participants of the user survey with larger screens found that the application did not scale well, and it resulted in a large amount of white space among other issues. Additionally, the mobile layout of the application is not as user-friendly as the desktop layout, particularly on the circuit builder page. Given that one of the project's original goals was accessibility, this is a significant limitation that would need to be addressed in future work (Section 6.5). Another mobile-related limitation is the lack of support for touchscreen events, meaning that the application is not accessible on mobile devices without a keyboard and mouse. Similarly, this is a limitation on the accessibility of the application that would need to be addressed in future work.

Another limitation of the project is the lack of advanced lessons in the educational resources. The lessons that were implemented were introductory and foundational, and while they were successful in educating the participants, more advanced lessons would be needed to provide a comprehensive educational experience. Additionally, as discussed in Section 6.1.3, the content in the educational resources was not as comprehensive as it could have been. The responses indicated that some participants may have struggled to understand the content in its entirety and that the assumed prior knowledge may have been too high for some participants.

Finally, a limitation of the project is that the quantum computations are not calculated by the application itself. Instead, the application uses the Cirq library to calculate the quantum operations. This is a limitation in the sense that the application is not a standalone quantum simulator, but rather a front-end for the Cirq library. This decision was driven by the original time limitation on the project, and the need to focus on the educational content of the application. Further research should be done to determine the feasibility of implementing a standalone quantum simulator in the future, and the advantages and disadvantages of doing so.

In summary, the primary limitations of the project are:

- The restricted responsive design of the application.
- The lack of support for touchscreen events.
- The lack of advanced lessons in the educational resources.
- The content in the educational resources was not as comprehensive as it could have been.

- The application is not a standalone quantum simulator.

6.5 Further Work

While the project has been successful in achieving its original goals, several areas could be improved upon in the future. Some of the original requirements that were not met should be addressed. The requirement concerning containerising the application to be deployed as a desktop application may not be within the scope of the project in the future, particularly given its current success as a web application. However, the three remaining unmet requirements could be addressed in future work.

The two requirements concerning composite gates would be a strong addition to the application, as they would improve the user experience, particularly for more intermediate to advanced users. However, due to the nature of the current implementation of the application, this would require a significant amount of work to implement — particularly on the circuit builder front-end.

To improve the application's educational value, further iterations of development could be very beneficial — where development is focused on enriching the lessons and composing new lessons. This should be supported by more rigorous user surveys at each iteration to receive detailed feedback. This would allow for a more comprehensive understanding of the educational content and how it could be improved. This could include questions such as:

- Did you find any of the concepts to be over or under-explained? If so, please specify.
- Are the quizzes too easy or too hard?
- Is there too much or too little content per lesson?
- From the sections you have completed, please select the section(s) you found most difficult to understand.

Additionally, extra iterations of development could be used to address the responsive design and touchscreen compatibility issues. This would involve implementing additional breakpoints in the application to ensure that it scales well on all devices and modifying the existing JavaScript modules concerning circuit building to accommodate touch events. Further efforts into the mobile usability of the application could also be made — for example, a large portion of the helpful information in the circuit builder relies on mouse hover events, which are not accessible on mobile devices. This could be addressed by implementing a 'long press' event to display the information on mobile devices instead.

Participants in the user feedback survey provided suggestions for improvements to the application (Section 6.1.4 and Table 6.2), which could also be addressed in future work. These include:

- Guided tutorials for constructing quantum circuits.

- Bloch sphere visualisation of qubits within a circuit (as opposed to only having Bloch sphere visualisations of gate transformations).
- The existing tutorial should be accessible from the builder page.
- A custom file extension for downloaded circuits.
- More visual aids in general.

Finally, in the future, research should be done regarding implementing a standalone quantum simulator in the application (Section 6.4). This would involve researching the advantages and disadvantages of doing so, and the potential impact on the performance, reliability, and accuracy of the application. This would be a significant undertaking and would require a substantial amount of research and development, but it could be a valuable addition to the application if it were successful.

6.5.1 Suggested Plan for Further Work

A set of requirements for further work has been compiled based on the unmet original requirements, limitations of the project and the feedback received in the user survey. These could be used to guide a '*second phase*' of development for the application. These are tentative and should be subject to refinement and change, and can be seen in Table 6.3.

ID	Requirement
R1	The application should have a dedicated design and layout for small screens.
R2	The application should have a dedicated design and layout for medium screens.
R3	The application should have a dedicated design and layout for large screens.
R4	The application should have a dedicated design and layout for extra large screens.
R5	The application should allow users to download and upload circuit files with a custom extension such as .qc rather than as a .json file.
R6	The circuit builder should allow touchscreen drag-and-drop functionality
R7	The circuit builder should provide useful information about gates upon a 'long-press' touchscreen touch.
R8	The circuit builder should provide a Bloch sphere state visualisation of a qubit within a circuit, at any point in a circuit.
R9	The circuit builder could have a moments walkthrough feature, that steps the user through each moment in the circuit, visualising qubit states at each moment.
R10	The circuit builder should allow a collection of simple gates to be compiled into a composite gate. (R9 in the original requirements)
R11	A user should be able to save and 're-use' a created composite gate. (R10 in the original requirements)
R12	The educational lessons should have dedicated quiz alerts, rather than browser defaults.
R13	The application could contain an additional series of interactive lessons that teach the user about more advanced concepts (R13 in the original requirements)
R14	The application could contain guided tutorials to building and understanding specific quantum circuits.

Table 6.3: Tentative requirements for potential future work.

Chapter 7

Conclusions

The original aim of this project was to create an educational platform with a quantum circuit simulator, that would be intuitive and user-friendly, for those with an undergraduate level of computer science knowledge. The project was driven by both the lack of quantum educational resources aimed at this specific demographic, and research showing that teaching through application rather than through theory can be more effective.

The platform was created as a web application, built using Django and Google's Cirq quantum library, with Tailwind CSS for styling, and successfully deployed with Heroku. The application features a quantum circuit simulator, with a drag-and-drop interface, and a set of educational resources, including a series of lessons, and a spotlight-style tutorial. The platform was tested with a group of undergraduate students, 59% of whom take a computer science-related degree, who were asked to complete a series of tasks on the platform and provide feedback.

Of the 14 original project requirements, 10 were met, with time constraints being the main challenge throughout. Despite this, the platform has been successful in fulfilling its aims, supported by the positive feedback received from the user survey, alongside the other testing completed — unit testing, user-acceptance testing, functional testing, and a Google Lighthouse audit. All of the participants in the user survey strongly agreed that they understood more about quantum computing concepts after using the platform, consolidating the project's success educationally.

The Google Lighthouse audit displays a minimum degree of accessibility, with a score of 100% for desktop and 99% for mobile. This is supported further by the positive feedback from the user survey regarding the platform's accessibility. The functional testing too was successful, demonstrating the platform's cross-browser compatibility. The unit testing was also a strong indicator of the platform's quantum correctness and general robustness, with 100% coverage in the back-end, and 89.32% in the front-end.

The project has a number of constraints and limitations. Constraints regarding the user survey, have been identified and addressed, such as the small sample size and recency bias. The limitations of the project lie particularly in its mobile compatibility, the lack of more advanced educational content, and in that the application is not a standalone quantum

simulator. Given more time, the project could have been improved by addressing these limitations, and by implementing the remainder of the original project requirements. Future work should include implementing mobile compatibility, enhancing the existing educational content, and implementing additional educational resources. Future research should focus on the feasibility of creating a standalone quantum simulator, and on identifying the advantages and disadvantages of this.

The original research conducted at the beginning of the project showed that quantum educational resources for undergraduates with a computer science background are limited. This project, supported by a diverse set of testing and success indicators, has successfully fulfilled its aims. The project has shown that targeted resources for computer science students can be created, are effective, and has provided a strong foundation for future research and work in this area.

Bibliography

[1] ApexCharts developers. ApexCharts – a modern JavaScript charting library, 2018. Version 3.30.

[2] Butscher, B., and Weimer, H. libquantum – the C library for quantum computing and quantum simulation, 2013. Version 1.1.1.

[3] Carrascal, G., del Barrio, A. A., and Botella, G. First experiences of teaching quantum computing. *The Journal of supercomputing 77*, 3 (2021), 2770–2799.

[4] Cirq Developers. Cirq – an open source framework for programming quantum computers, 2022. Version 1.3.0.

[5] Cohen, D. On holy wars and a plea for peace. *Computer (Long Beach, Calif.) 14*, 10 (1981), 48–54.

[6] Devopedia. 'Qubit' – arbitrarily manipluable two-state quantum system, Jan. 2022.

[7] Dirac, P. A. M. A new notation for quantum mechanics. *Mathematical proceedings of the Cambridge Philosophical Society 35*, 3 (1939), 416–418.

[8] Django developers. Django – the web framework for perfectionists with deadlines, 2005. Version 4.0.

[9] Dumon, K. The computational power of Quantum Computers: an intuitive guide, Apr. 2019.

[10] Facebook, Inc. React - the library for web and native user interfaces, 2013.

[11] Fingerhuth, M., Babej, T., and Wittek, P. Open source software in quantum computing, 2018.

[12] Flask developers. Flask – a micro web framework for Python, 2010. Version 2.1.

[13] Gerdt, V., and Prokopenya, A. Simulation of quantum error correction with mathematica. *Lecture Notes in Computer Science 8136* (09 2013), 116–129.

[14] Gerjuoy, E. Shor's factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers. *American Journal of Physics 73*, 6 (06 2005), 521–540.

[15] Gidney, C. Quirk's performance history, 2016.

[16] Gidney, C. Quirk - a drag-and-drop quantum circuit simulator, 2023.

[17] Google LLC. Angular - the web development framework for building the future, 2010.

[18] Google LLC. Lighthouse - an open-source, automated tool for improving the quality of web pages, 2016.

[19] Heroku developers. Heroku – a cloud platform as a service supporting several programming languages, 2007.

[20] IBM developers. IBM Circuit Composer, 2023.

[21] Jest developers. Jest – a delightful JavaScript testing framework, 2014.

[22] Joanna Patrzyk and Bartłomiej Patrzyk. QuIDE - a quantum computer simulation platform written in C#, 2014.

[23] Kumar, P., and Patel, A. Quantum error correction using weak measurements. *Quantum Information Processing 18* (01 2019).

[24] Microsoft developers. *Quantum Development Kit*. Microsoft Corporation, 2023.

[25] Mozilla developers. Canvas API – a powerful, low-level 2D rendering API for the web, 2004.

[26] Mozilla developers. Fetch API – a modern interface for fetching resources, 2015.

[27] Mykhailova, M., and Svore, K. M. Teaching quantum computing through a practical software-driven approach: Experience report. In *Proceedings of the Conference* (Ithaca, 2020), Cornell University Library, arXiv.org.

[28] Nielsen, M. A., and Chuang, I. L. *Quantum computation and quantum information*. Cambridge University Press, Cambridge, 2000.

[29] Oracle Corporation. JavaFX – open source, next generation client application platform for desktop, mobile and embedded systems built on Java, 2008.

[30] Pytest developers. Pytest - helps you write better programs, 2003.

[31] Python Software Foundation. Tkinter - the Python interface for Tk, 1991.

[32] Qiskit Developers. Qiskit – an open-source framework for quantum computing, 2017. Version 0.45.0.

[33] Rails developers. Ruby on Rails – a web-application framework, 2004. Version 6.1.

[34] Riverbank Computing Limited. PyQt – a Python binding for Qt, 1998. Version v5.

[35] Roffe, J. Quantum error correction : an introductory guide, 2019.

[36] Rycerz, K., Patrzyk, J., Patrzyk, B., and Bubak, M. Teaching quantum computing with the quide simulator. *Procedia Computer Science 51* (2015), 1724–1733. International Conference On Computational Science, ICCS 2015.

[37] Shor, P. W. Scheme for reducing decoherence in quantum computer memory. *Physical review. A, Atomic, molecular, and optical physics 52*, 4 (1995), R2493–R2496.

[38] Tailwind CSS developers. Tailwind CSS – a utility-first CSS framework for rapidly building custom designs, 2017. Version 3.0.

[39] Testing Library developers. jest-dom – custom Jest matchers to test the state of the DOM, 2018.

[40] UK, Q. C. Quantum Error Correction: Phase flip code in Qiskit, Aug. 2020.

[41] Vue.js developers. Vue.js – The Progressive JavaScript Framework, 2014. Version v3.0.

[42] Webpack developers. Webpack – a static module bundler for modern JavaScript applications, 2012. Version 5.0.

[43] Wong, T. G. *Introduction to Classical and Quantum Computing*. Rooted Grove, 2022.

[44] Wootters, W. K., and Zurek, W. H. A single quantum cannot be cloned. *Nature 299* (1982), 802–803.

Appendices

Appendix A

Destop Website Images

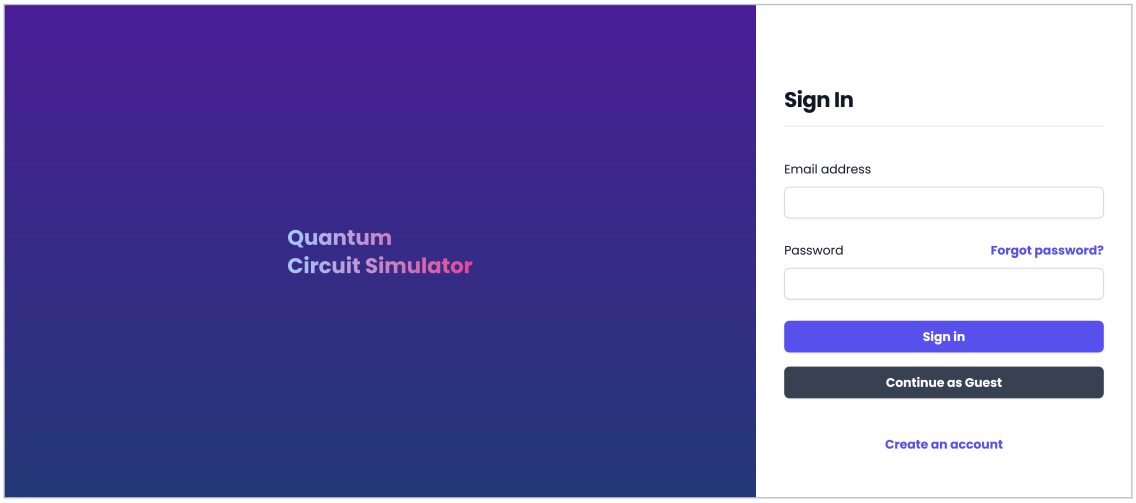


Figure A.1: Desktop login page.

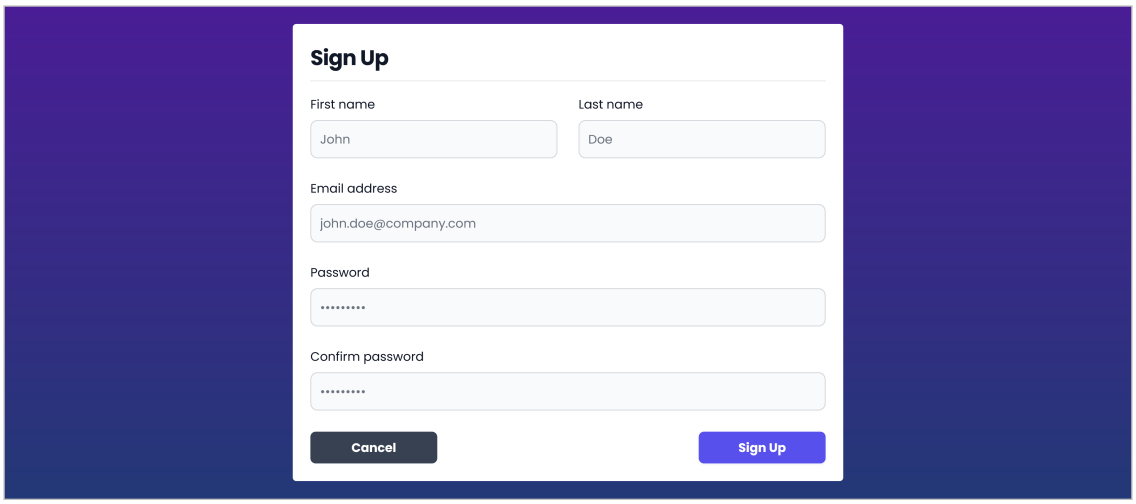


Figure A.2: Desktop sign-up page.

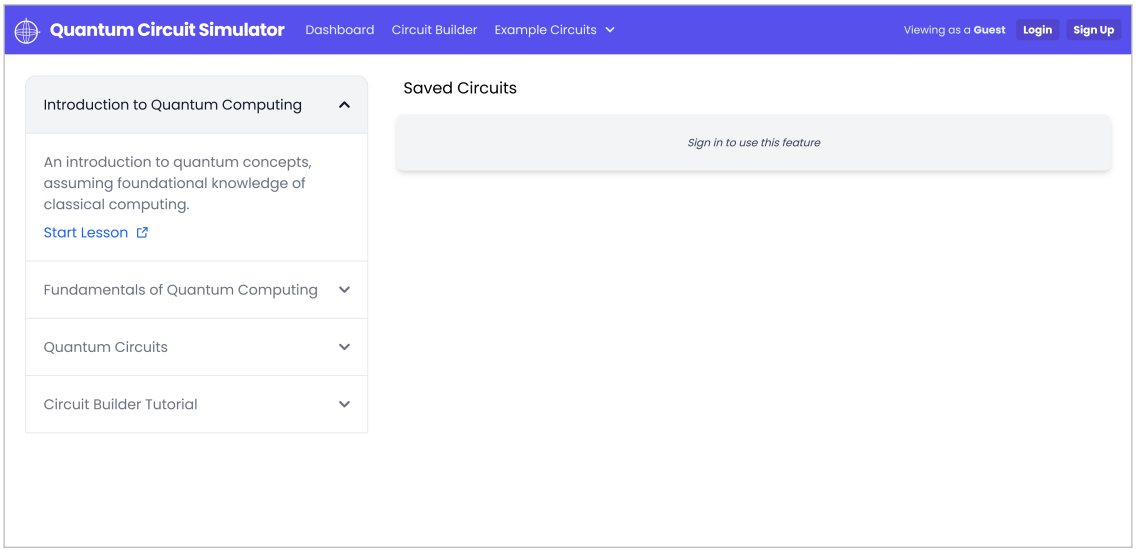


Figure A.3: Desktop guest dashboard page.

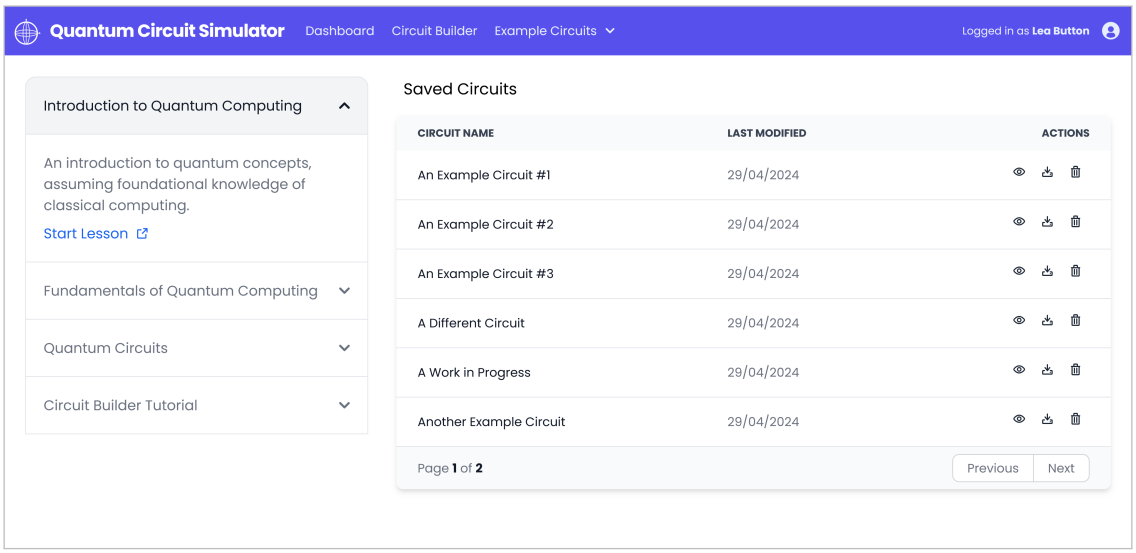


Figure A.4: Desktop logged-in dashboard page.

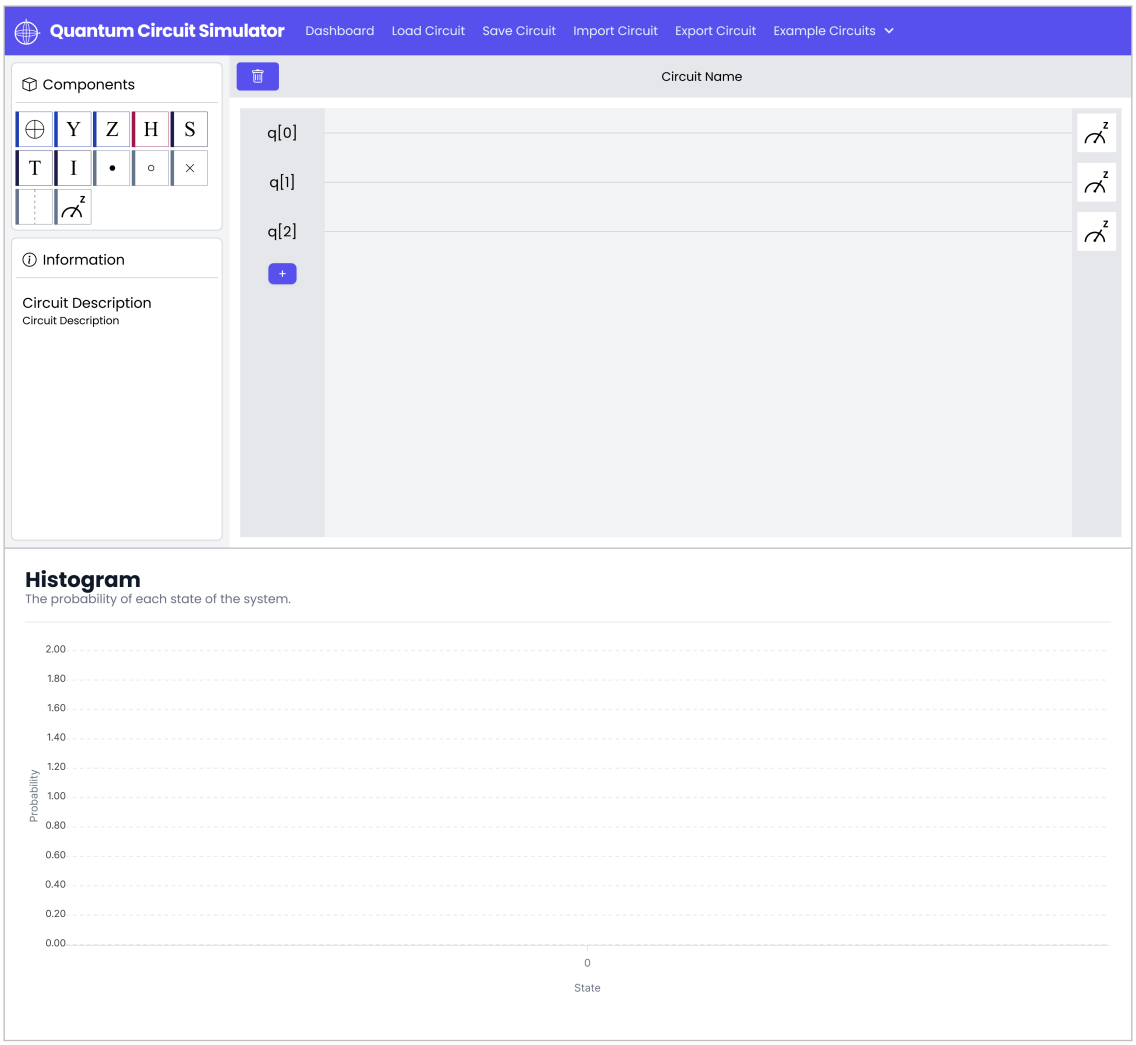


Figure A.5: Desktop empty circuit builder page.

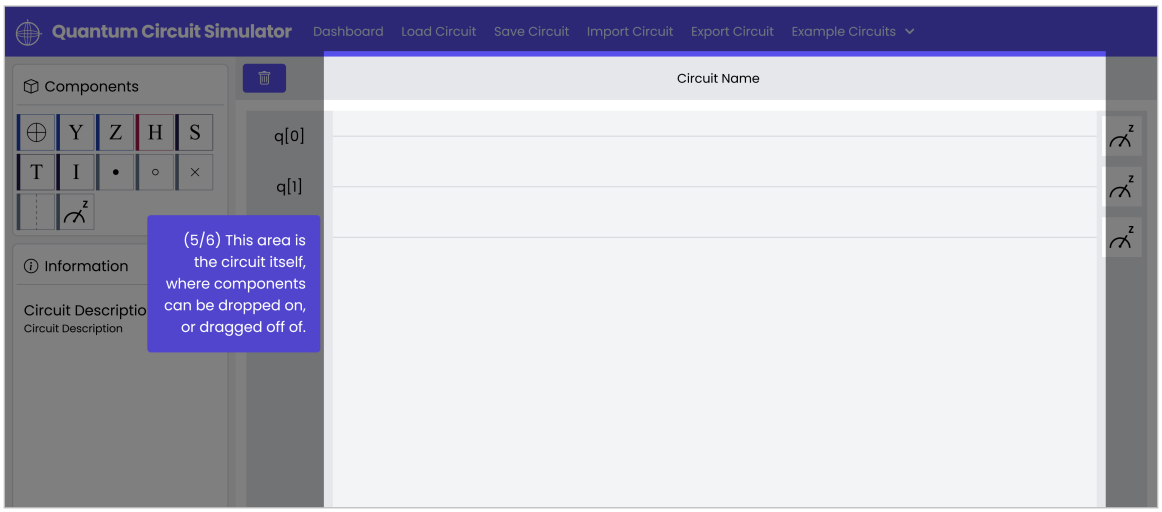


Figure A.6: Desktop circuit builder tutorial page.

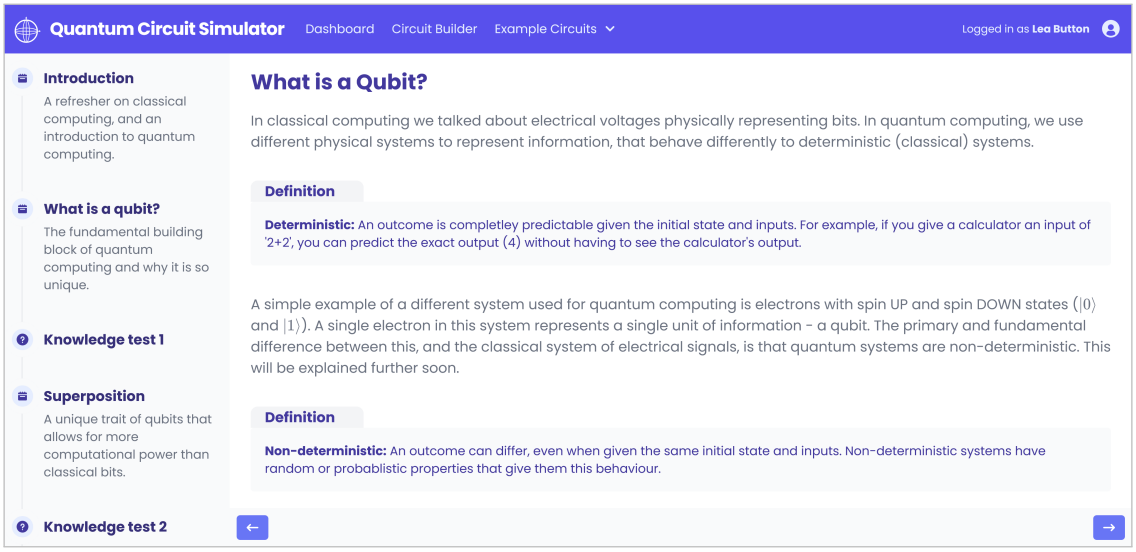


Figure A.7: Desktop lesson snapshot of the 'What is a Qubit' section.

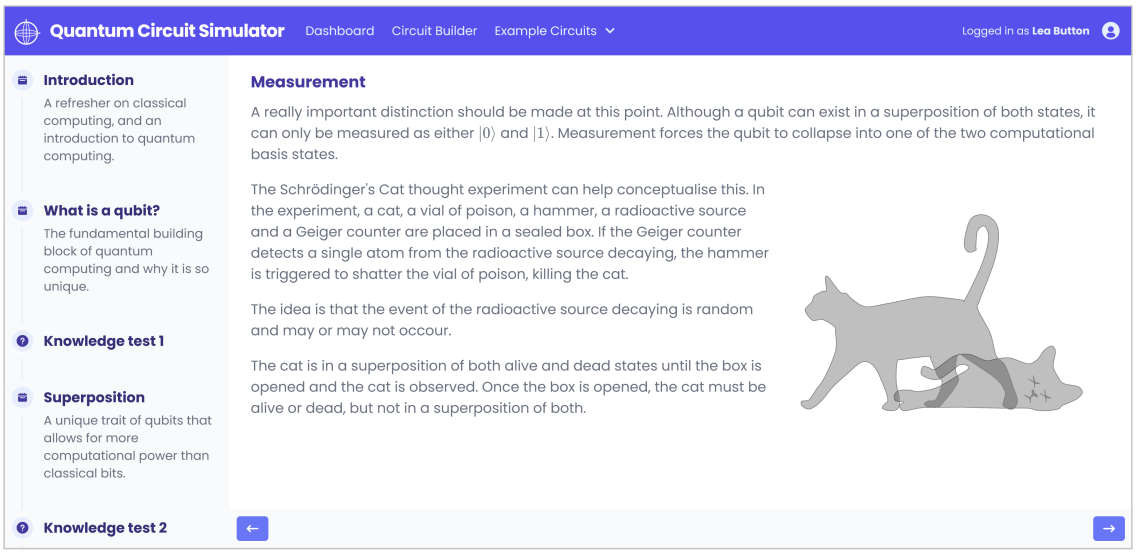


Figure A.8: Desktop lesson snapshot of the 'Measurement' section.

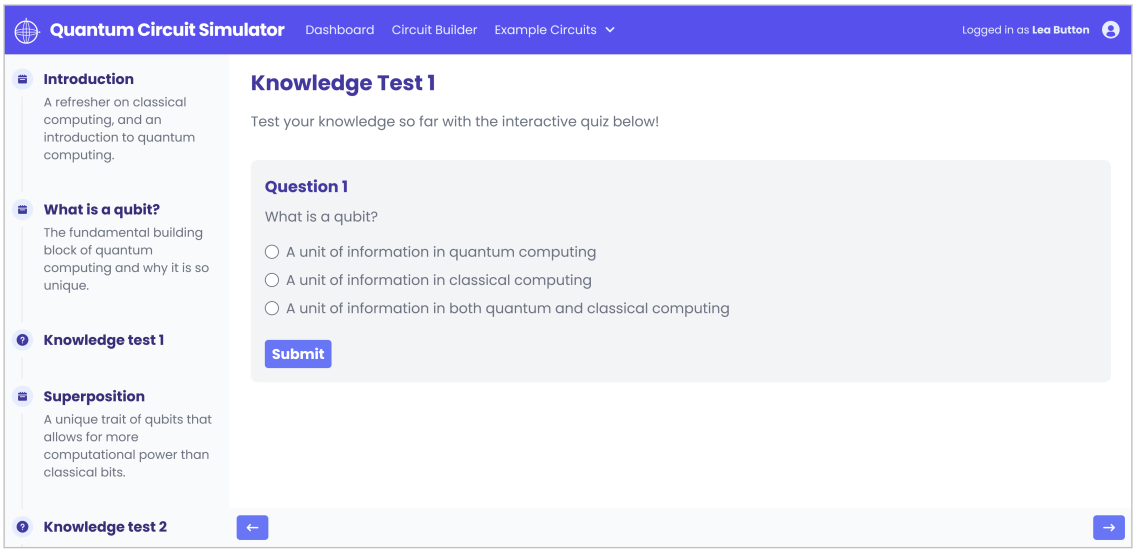


Figure A.9: Desktop interactive lesson quiz example.

Appendix B

Mobile Website Images

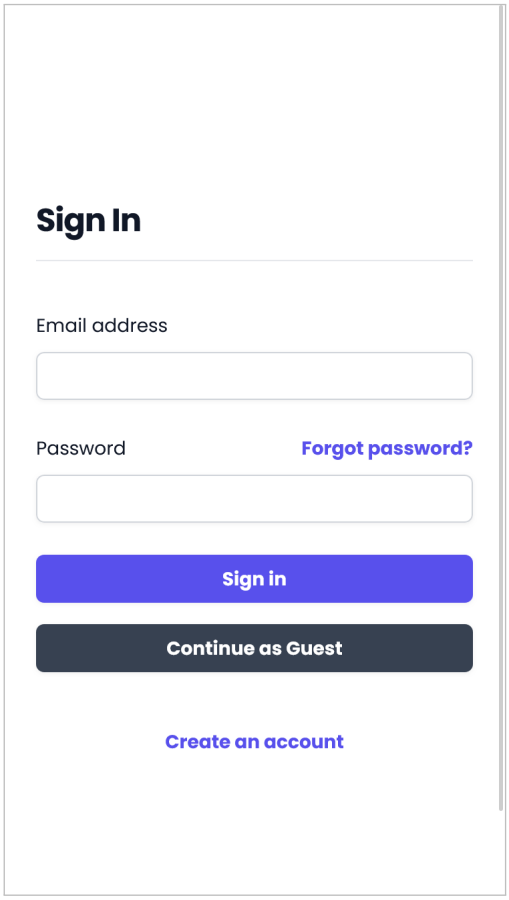


Figure B.1: Mobile login page.

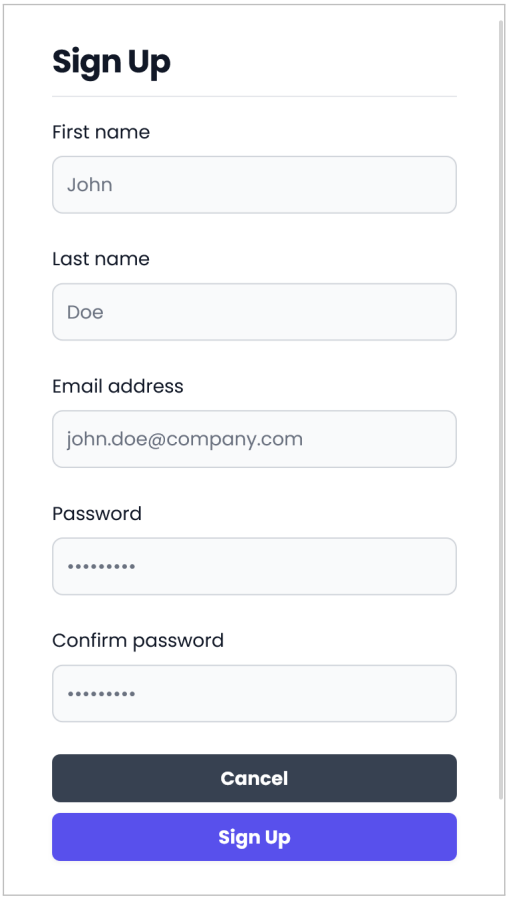


Figure B.2: Mobile sign-up page.

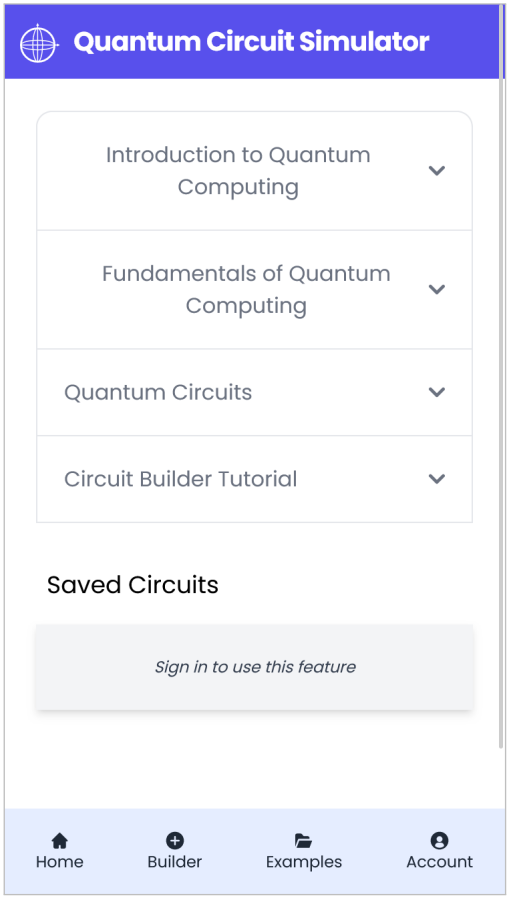


Figure B.3: Mobile guest dashboard page.

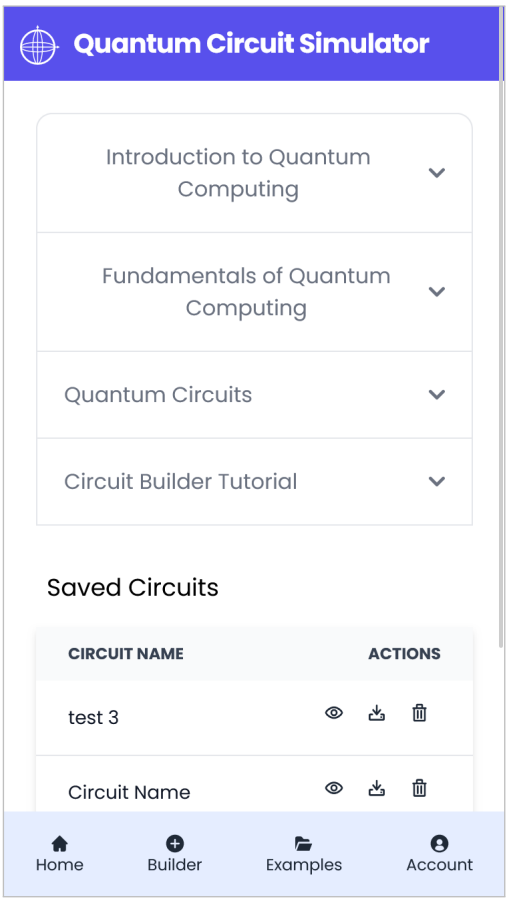


Figure B.4: Mobile logged-in dashboard page.

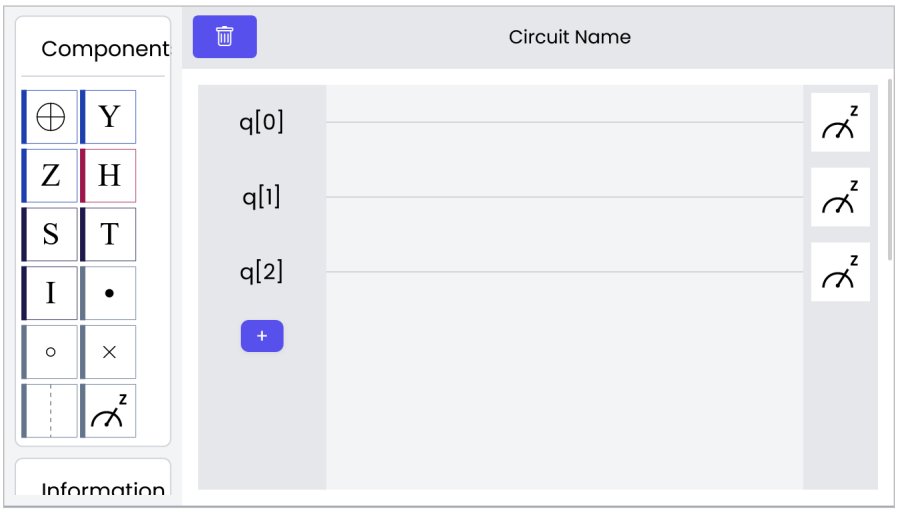


Figure B.5: Mobile empty circuit builder page.

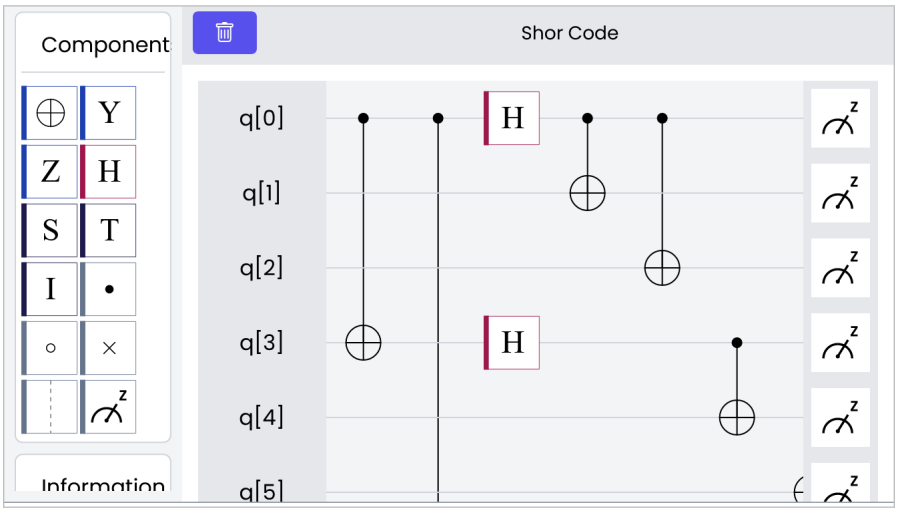


Figure B.6: Mobile circuit builder (partially) displaying Shor Code.

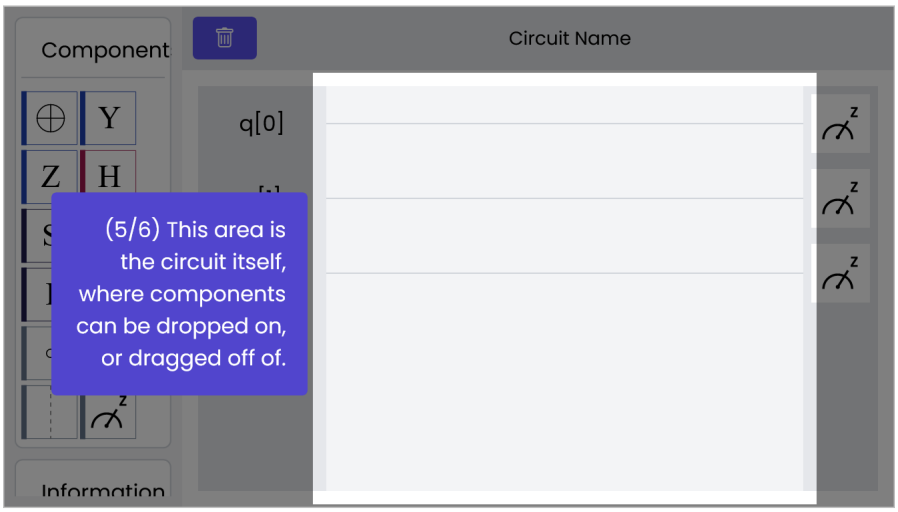


Figure B.7: Mobile circuit builder tutorial page.

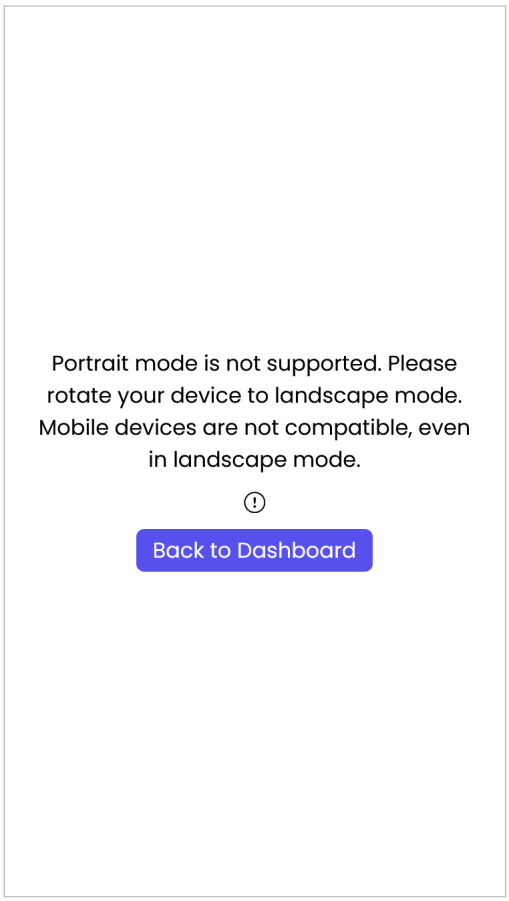


Figure B.8: Mobile portrait circuit builder.

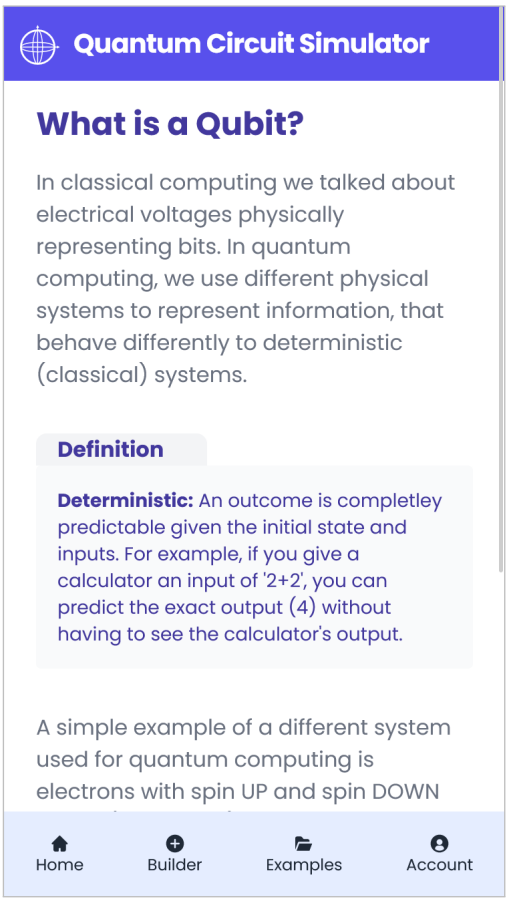


Figure B.9: Mobile lesson snapshot of the 'What is a Qubit' section.

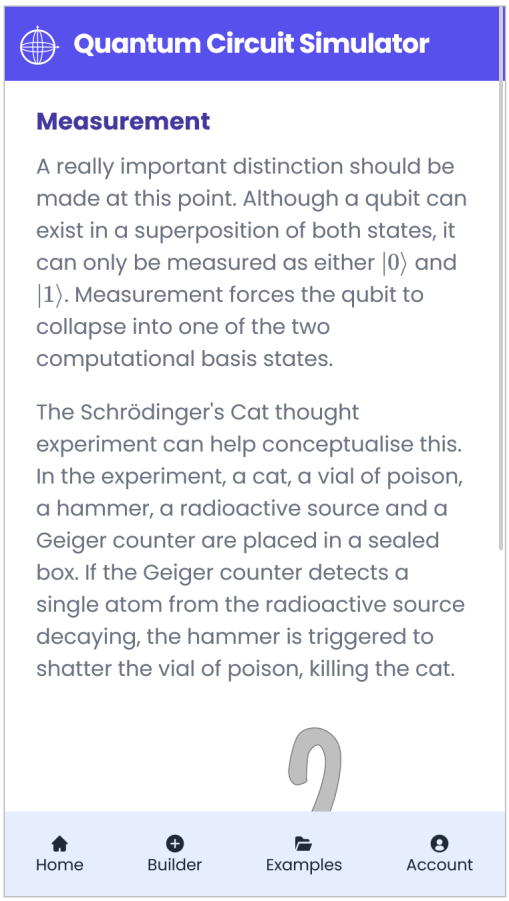


Figure B.10: Mobile lesson snapshot of the 'Measurement' section.

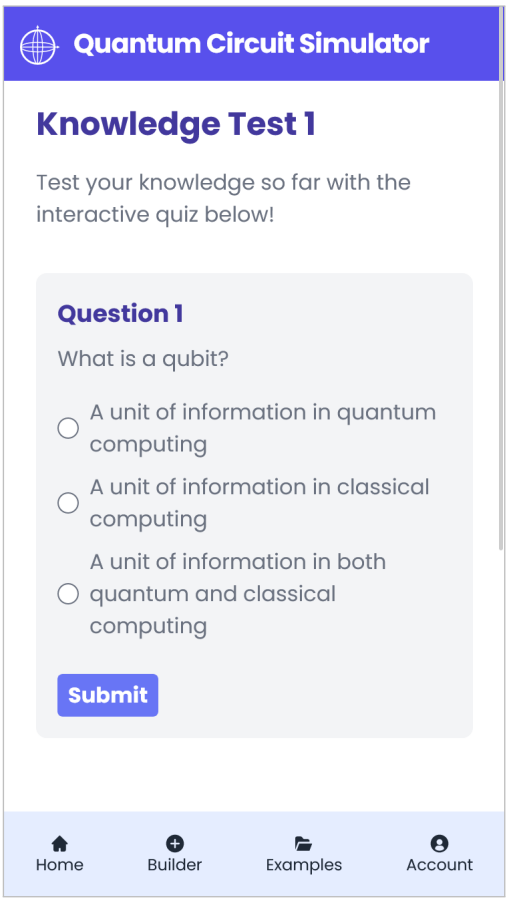


Figure B.11: Mobile interactive lesson quiz example.

Appendix C

Ethics Approval

Downloaded: 28/04/2024
Approved: 12/04/2024

Lea Button
Registration number: 200337856
Computer Science
Programme: Computer Science BSc

Dear Lea

PROJECT TITLE: A Quantum Circuit Simulator
APPLICATION: Reference Number 059641

On behalf of the University ethics reviewers who reviewed your project, I am pleased to inform you that on 12/04/2024 the above-named project was **approved** on ethics grounds, on the basis that you will adhere to the following documentation that you submitted for ethics review:

- University research ethics application form 059641 (form submission date: 27/03/2024); (expected project end date: 08/05/2024).
- Participant information sheet 1135325 version 1 (27/03/2024).
- Participant consent form 1135326 version 1 (27/03/2024).

If during the course of the project you need to deviate significantly from the above-approved documentation please inform me since written approval will be required.

Your responsibilities in delivering this research project are set out at the end of this letter.

Yours sincerely

Luke Whitham
Ethics Administrator
Computer Science

Please note the following responsibilities of the researcher in delivering the research project:

- The project must abide by the University's Research Ethics Policy: https://www.sheffield.ac.uk/research-services/ethics-integrity/policy
- The project must abide by the University's Good Research & Innovation Practices Policy: https://www.sheffield.ac.uk/polopoly_fs/1.671066!/file/GRIPPolicy.pdf
- The researcher must inform their supervisor (in the case of a student) or Ethics Administrator (in the case of a member of staff) of any significant changes to the project or the approved documentation.
- The researcher must comply with the requirements of the law and relevant guidelines relating to security and confidentiality of personal data.
- The researcher is responsible for effectively managing the data collected both during and after the end of the project in line with best practice, and any relevant legislative, regulatory or contractual requirements.

Appendix D

Participant Consent Form

A Quantum Circuit Simulator: Participant Consent Form

Taking part in the project

I have read and understood the project information sheet. (If you answer No, do not proceed with this consent form until you are fully aware of what your participation in the project will mean.)	
I have been given the opportunity to ask questions about the project.	
I agree to take part in the project. I understand that participating in the survey involves providing my opinions and some demographic information.	
I understand that choosing to participate as a volunteer in this research, does not create a legally binding agreement, nor is it intended to create an employment relationship with the University of Sheffield	
I understand that my participation is voluntary and that I can withdraw from the study at any time. I do not have to give any reasons for why I no longer want to participate, and there will be no adverse consequences if I choose to withdraw.	

How my information will be used during and after the project

I understand that all the information provided will be anonymised.	
I understand and agree that the authorised researchers will have access to this data and may use the data in publications, reports, web pages, and other research outputs only if they agree to preserve the confidentiality of the information as requested in this form.	

So that the information you provide can be used legally by the researchers

I agree to assign the copyright I hold in any materials generated as part of this project to The University of Sheffield	

Appendix E

Participant Information Sheet

A Quantum Circuit Simulator: Information Sheet

You are being invited to take part in a research project. Before you decide whether or not to participate, it is important for you to understand why the research is being done and what it will involve. Please take time to read the following information carefully and discuss it with others if you wish. Ask us if there is anything that is not clear or if you would like more information. Take time to decide whether or not you wish to take part. Thank you for reading this.

(1) What is the project's purpose?

The purpose of this project is to create a web-based educational quantum circuit simulator, specifically aimed at those with an undergraduate or higher level of computer science understanding. This survey aims to aid in the evaluation of this project, in its success as an educational and user-friendly tool.

(2) Why have I been chosen?

You have been chosen because you are either currently studying, hold an undergraduate degree, or otherwise work in a relevant field.

Note: you may not be studying or working within a computer science related field, in which case, your feedback on the design and structure of the platform is still valued.

(3) Do I have to take part?

Your participation in the project is completely voluntary. If you do decide to take part you will be asked to complete a consent form within the Google Form (and you can also request a copy of this information sheet by contacting the primary researcher Lea Button at lgbutton1@sheffield.ac.uk) and you will be automatically directed to the study which will involve answering questions about the design, layout and usability of the platform. Even if you give consent, you are free to withdraw from the study at any time with no consequences and without giving a reason by simply closing the web browser on your smartphone, tablet or computer, and your responses will not be recorded. Please note that by choosing to participate in this research, this will not create a legally binding agreement, nor is it intended to create an employment relationship between you and the University of Sheffield.

(4) What will happen to me if I take part? What do I have to do?

The survey should take approximately 10-45 minutes, depending on how much you choose to answer. The survey has no time limit, and you may exit and return to the survey as you please. You will first be asked for some demographic information, then you may choose to complete a shorter (option A), or longer survey (option B).

If you choose option A the survey will follow the following structure:

1. Providing feedback on the circuit building and visualisations on the platform.
2. (Optional) additional comments.

If you choose option B the survey will follow the following structure:

1. Providing feedback on the educational walkthrough(s) on the platform.
2. Providing feedback on the circuit building and visualisations on the platform.
3. (Optional) additional comments.

Finally, you may withdraw consent, and exit the survey entirely, without providing a reason, at any time.

(5) What are the possible disadvantages and risks of taking part?

There is a risk that you are inconvenienced by the time taken to complete the survey. Additionally, there is a risk that upon choosing to complete the longer survey, you dislike or find boring the educational content of the platform.

(6) What are the possible benefits of taking part?

(7) Will my taking part in this project be kept confidential?

Yes. Your data will be anonymised, and you will not be personally identifiable in any reports, documentation or similar.

(8) What is the legal basis for processing my personal data?

According to the data protection legislation, we are required to inform you that the legal basis we are applying in order to process your personal data is that 'processing is necessary for the performance of a task carried out in the public interest' (Article 6(1)(e)).

Further information can be found in the University's Privacy Practice Notice
https://www.sheffield.ac.uk/govern/data-protection/privacy/general

(9) What will happen to the data collected, and the results of the research project?

The data will be analysed by Lea Button. During data collection, your responses will be stored fully anonymised, without any identifiable personal data. The data will be destroyed within 3 months of the project's completion.

(10) Who is organising and funding the research?

The research is organised by Lea Button, the primary and sole researcher, and it is not funded.

(11) Who is the Data Controller?

The University of Sheffield will act as the Data Controller for this study. This means that the University is responsible for looking after your information and using it properly.

(12) Who has ethically reviewed the project?

This project has been ethically approved via the University of Sheffield's Ethics Review Procedure, as administered by the Computer Science department. The ethics approval number is 059641.

(13) What if something goes wrong and I wish to complain about the research or report a concern or incident?

If you are dissatisfied with any aspect of the research and wish to make a complaint, please contact the researcher (Lea Button: lgbutton1@sheffied.ac.uk) or the project supervisor (Phil Mcminn: p.mcminn@sheffield.ac.uk) in the first instance. If you feel your complaint has not been handled in a satisfactory way, you can contact the Head of the Department of Computer Science, Prof. Heidi Christensen: heidi.christensen@sheffield.ac.uk. If the complaint relates to how your personal data has been handled, you can find information about how to raise a complaint in the University's Privacy Notice: https://www.sheffield.ac.uk/govern/data-protection/privacy/general.

(14) Contact for further information

If you have any questions either ahead of participation or relating to your data and the study, please contact the researcher (Lea Button: lgbutton1@sheffied.ac.uk). An electronic copy of this information sheet and consent form can be provided on request by contacting the researcher(Lea Button: lgbutton1@sheffied.ac.uk).

Appendix F

Survey Questions

An Educational Quantum Circuit Builder and Simulator: User Survey

This survey aims to collect user feedback on a web-based educational quantum simulator.

The platform is best accessed through a desktop device - we do not recomend accessing the site on mobile. The survey should take an estimated time of 15-35 minutes.

The platform involves both a circuit building and simulation area and various walk-through lessons to teach users about quantum circuit building. Many quantum circuit simulators exist already; however, many are very verbose and intimidating for beginners to use. Additionally, there are very few resources for those with a computer science background to learn quantum computing - most resources assume either a physics background or no background. With these two points in mind, the platform will be developed in a beginner-friendly way and aimed at those with an undergraduate (or higher) level of computer science knowledge.

Depending on your preference, you will complete either a long or short option of the survey, answering questions about your opinion on the platform. No personal data will be collected from this survey.

Copies of the project information sheet and the participant consent form can be found: here https://docs.google.com/document/d/1121yM_NTXzZQxZy3O9RvC1j7vR7iZvs9/edit?usp=sharing&ouid=115234115302746659670&rtpof=true&sd=true and here https://docs.google.com/document/d/1NCtXcv4_JQB3an0_uCPW61532B7sqXDA/edit?usp=sharing&ouid=115234115302746659670&rtpof=true&sd=true respectively.

If you wish to report any issues, or ask any questions, please do not hesitate to contact the primary researcher Lea Button at lgbutton1@sheffield.ac.uk.

* Indicates required question

1. I understand what this survey entails, how my data will be used and wish to proceed. *

   Mark only one oval.

   - ◯ Yes, I would like to proceed.
   - ◯ No, I would not like to proceed.

Consent Form

Informed consent must be collected from you (the participant) before continuing with the survey.

2. I have read and understood the project information sheet. (If you answer No, do not proceed with this consent form until you are fully aware of what your participation in the project will mean.) *

   Mark only one oval.

   - ◯ Yes
   - ◯ No

3. I have been given the opportunity to ask questions about the project. *

   Mark only one oval.

   - ◯ Yes
   - ◯ No

4. I agree to take part in the project. I understand that participating in the survey involves providing my opinions and some demographic information. *

   Mark only one oval.

   - ◯ Yes
   - ◯ No

5. I understand that choosing to participate as a volunteer in this research, does not create a legally binding agreement, nor is it intended to create an employment relationship with the University of Sheffield *

   Mark only one oval.

   - ◯ Yes
   - ◯ No

6. I understand that my participation is voluntary and that I can withdraw from the study at any time. I do not have to give any reasons for why I no longer want to participate, and there will be no adverse consequences if I choose to withdraw. *

   Mark only one oval.

   - ◯ Yes
   - ◯ No

7. I understand that all the information provided will be anonymised. *

   Mark only one oval.

   - ◯ Yes
   - ◯ No

8. I understand and agree that the authorised researchers will have access to this data and may use the data in publications, reports, web pages, and other research outputs only if they agree to preserve the confidentiality of the information as requested in this form. *

   Mark only one oval.

   - ◯ Yes
   - ◯ No

9. I agree to assign the copyright I hold in any materials generated as part of this project to The University of Sheffield *

Mark only one oval.

- ◯ Yes
- ◯ No

Demographic Information

The next section will ask some demographic questions.

10. What gender do you identify as? *

Mark only one oval.

- ◯ Female
- ◯ Male
- ◯ Non-binary
- ◯ Prefer not to say
- ◯ Other: _____

11. What age group do you belong to? *

Mark only one oval.

- ◯ 18 - 24
- ◯ 25 - 34
- ◯ 35 - 44
- ◯ 45 - 54
- ◯ 55 - 64
- ◯ 65+

12. I am: *

Mark only one oval.

- ◯ An undergraduate student
- ◯ A postgraduate student
- ◯ Employed (part-time or full-time)
- ◯ Other: _____

13. Do you study or work in a computer science related field? *

Mark only one oval.

- ◯ Yes
- ◯ No

14. Before this survey, how much prior knowledge of quantum computing, specifically quantum circuits, do you have? *

Mark only one oval.

- ◯ No prior knowledge
- ◯ Basic understanding
- ◯ Moderate understanding
- ◯ Advanced understanding

Part 1 (all participants)

The site can be accessed here: https://edu-quantum-simulator-02a0eea10e84.herokuapp.com/

Please navigate to the site and choose to continue either as a guest, or sign up and log in.

Feel free to explore the platform, however this section is concerned specifically with the 'circuit builder' page of the site. In the circuit builder, either open an example circuit (located in the top navigation bar), or create your own. (If you are unsure, we recomend opening either the bit-flip, or phase-flip code examples!)

Once you feel you have had enough time to navigate around the circuit builder, please answer the following questions on a scale of 1-5. 1 being strongly disagree, and 5 being strongly agree.

15. Selecting an example circuit or building my own was easy. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

16. I could easily find all the tools or functions I needed. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

17. The overall design of the circuit builder is simple and easy to navigate *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

18. The circuit visualisation is easy to understand. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

19. The probability visualisations are easy to understand. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

20. The loading times for the platform are reasonable. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

Section 2 (optional)

If you wish to dedicate more time to this survey, we would like you to complete the 'Introduction to Quantum Computing' lesson and give your feedback on the content and quality.

21. I will complete the 'Introduction to Quantum Computing' lesson and I am willing to give additional feedback on this. *

Mark only one oval.

◯ Yes

◯ No *Skip to question 27*

Section 2 (optional)

The site can be accessed here: https://edu-quantum-simulator-02a0eea10e84.herokuapp.com/

Please sign up and log in, or continue as a guest and navigate to the dashboard. Here you can choose to begin the 'Introduction to Quantum Computing' lesson. Complete this at your own pace and return to this survey when you are ready.

Please answer the following questions on a
scale of 1-5. 1 being strongly disagree, and 5 being strongly agree.

22. The content of the teaching was engaging. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

23. The content of the teaching was easy to understand. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

24. The illustrated examples helped reinforce quantum concepts. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

25. The assumed prior knowledge was at the correct level for me. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

26. I understand more about quantum computing concepts now than I did prior to this experience. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly agree

Section 3 (optional)

This section is an optional opportunity to provide us with addtional feedback on the platform.

27. What did you like the most about the platform?

28. What improvements would you suggest for the platform?

29. Any other comments?

Thank you for participating in this survey.

Appendix G

Survey Free-text Responses Used for Coding

id	Response
r1	It was simple and **[easy to use]**$_{(c1)}$, the **[learning content was at the right level for me]**$_{(c4)}$ as a beginner.
r2	The short quizzes interspersed throughout the lessons helped solidify what was being taught.
r3	**[Engaging]**$_{(c2)}$ and **[easy to read and understand]**$_{(c3)}$.
r4	**[language was kept simple]**$_{(c3)}$ and terminology was explained which made the platform accessible
r5	All around **[pleasant look, nice graphics]**$_{(c5)}$
r6	**[good design]**$_{(c5)}$, content split up well
r7	The **[content was clear]**$_{(c3)}$ and **[easily understandable]**$_{(c3)}$. Everything was laid out in an **[easy to follow]**$_{(c1)}$ manner and I had **[no issues navigating]**$_{(c1)}$. The load times were fast and I did not encounter any bugs or issues. The example circuits and lessons were especially helpful in learning what I was looking at and uses of quantum computing.
r8	It was a great introduction to quantum computing given my incredibly rudimentary understanding of the subject, and all the **[content was mostly easy to understand]**$_{(c3)}$ given a history of computer science and mathematics
r9	The lessons, and the **[design & layout of the site]**$_{(c5)}$
r10	Website is **[very well designed]**$_{(c5)}$ and **[easy to use]**$_{(c1)}$ **[Lessons engaging]**$_{(c2)}$
r11	**[pretty]**$_{(c5)}$
r12	**[Nice fonts]**$_{(c5)}$ and **[clear topic sentences]**$_{(c3)}$.
r13	The website is very intuitive and **[easy to follow]**$_{(c1)}$
r14	**[simple and clean design]**$_{(c5)}$. **[Easy to navigate]**$_{(c1)}$

Table G.1: A table containing the responses to the free-text question 'What did you like the least about the platform?', highlighting the extracts used for coding.

id	Response
r1	in the lessons the next arrow was **[a bit hard to find]**$_{(c4)}$
r2	Maybe a few more/harder questions in the knowledge tests would make sure every aspect of the lesson is covered in a challenging format.
r3	guided tutorial exercises for building quantum circuits
r4	Some numbers in the bloch sphere illustrations were slightly out of place and if the **[correct/incorrect popup in the quizzes could be integrated into the website it would feel nicer]**$_{(c2)}$
r5	The **[pop ups when answering a question did not fit the theme]**$_{(c2)}$ of the website and looked unfinished, however they did function perfectly fine.
r6	Apart from **[one incorrect quiz answer about Hadamard gate]**$_{(c1)}$, some kind of visualisation of the possible states between the gates. e.g. if you hover over the gaps between moments, it would be good to be able to see the current super position of the bit. Also maybe **[make it clear that the histogram is below the circuit builder itself]**$_{(c4)}$
r7	None, but I think the **[correct/incorrect text should probably not appear as an alert]**$_{(c2)}$ (very nitpicky ik)
r8	There is **[a lot of white space]**$_{(c3)}$ on the Lessons, The next page button is small and **[have to scroll down to see it]**$_{(c4)}$ The Histogram is HUGE Circuit builder tutorial should be on the circuit builder page, **[only found it at the end]**$_{(c4)}$. Dashboard is **[90% white space]**$_{(c3)}$ Maybe give a custom file extension for downloaded circuits i.e. "Example.qc" (These are only small things, the website as a whole is very good)
r9	**[there's a lot of empty space]**$_{(c3)}$
r10	1. The saved circuits section seems to take up a very large part of the dashboard. **[When looking for the lesson it confused me a little]**$_{(c4)}$. 2. I personally **[do not like the dialogue boxes that say correct for the quizzes]**$_{(c2)}$, however I am not sure if there is a better way to implement this. 3. I think maybe have 3 questions per quiz. I like when things come in threes.
r11	Maybe more visual aids for those with very little CS background

Table G.2: A table containing the responses to the free-text question 'What improvements would you suggest for the platform?', highlighting the extracts used for coding.