Software Requirements Specification (SRS)

for

QThink: An Agentic Quantum Co-Pilot

Team QThink

October 27, 2025

Contents

1 Introduction

1.1 Purpose

This document specifies the requirements for **QThink**, an AI agentic assistant designed to accelerate the quantum computing research and development lifecycle. It serves as the primary guide for developers, testers, and project stakeholders throughout the 48-hour hackathon development period.

1.2 Scope

The system will provide a web-based interface for quantum researchers and developers to generate, debug, optimize, and simulate quantum algorithms. QThink will leverage the K2 Think API for advanced mathematical reasoning and a Context-Aided Generation (CAG) model for fast knowledge retrieval. To ensure maximum token efficiency for LLM interaction, the system will use a novel, domain-specific data format called **QuYAML**. The initial MVP will focus on Qiskit code generation and simulation, with a framework for a self-improving feedback loop.

1.3 Target Audience

The intended users are quantum computing researchers, students, and developers who require an intelligent tool to streamline their workflow.

2 Functional Requirements

This section provides a high-level summary of the system's core capabilities. Detailed scenarios are specified in Section 3: Use Case Specifications.

FR-1: User Input The system shall provide an interface for users to submit natural language queries, existing quantum code, or circuit definitions using the QuYAML format.

FR-2: Query Processing The system's agentic backend shall classify the user's intent (e.g., code generation, debugging, simulation, QuYAML parsing).

FR-3: Code Generation The system shall translate natural language descriptions or other formats into valid, executable Qiskit code.

FR-4: Code Optimization The system shall analyze provided Qiskit code and return an optimized version, including for specific hardware topologies.

FR-5: Mathematical Simulation The system shall perform a mathematical simulation of a circuit and display the final quantum state vectors and outcome probabilities.

FR-6: External Verification The system shall be capable of executing generated code on an external simulator (e.g., IBM Quantum) to verify real-world results.

FR-7: Feedback Logging The system shall log user interactions and results to a database to serve as the foundation for future self-improvement.

3 Use Case Specifications

This section provides detailed user stories and scenarios for the key functionalities of the QThink system.

3.1 Use Case #1 - High-Level VQE Implementation

User Story: 1 - High-Level VQE Implementation	
User Story ID	US-#1
User Story Name	High-Level VQE Implementation
Actors	Quantum Chemist
Description	As a **Quantum Chemist**, I want to **describe a molecule and a desired basis set**, so that I can **generate a complete Variational Quantum Eigensolver (VQE) circuit to find its ground state energy**.

Pre-conditions

- The user has knowledge of the molecular structure and the desired basis set.

Post-conditions

- The system generates a complete, executable Qiskit script and the equivalent QuYAML definition.

Acceptance Criteria

Given I am a quantum chemist, when I provide the molecular structure for LiH and specify the 'STO-3G' basis set, then the system generates a Qiskit script that correctly defines the qubit Hamiltonian, constructs a UCCSD ansatz, and sets up the VQE instance.

Scenarios

Normal Scenario	
Actor Action	**System Response**
User inputs: "Generate a VQE for Lithium Hydride at 0.735 angstrom using a UCCSD ansatz and STO-3G basis."	1. System parses the request and identifies the molecule, parameters, and algorithm. 2. System uses its CAG to retrieve standard parameters for LiH and the UCCSD ansatz structure. 3. System calculates the fermionic Hamiltonian and performs a Jordan-Wigner transformation. 4. System generates the Python Qiskit script and the equivalent QuYAML file. 5. System presents the generated files and a summary of the mapping to the user.

3.2 Use Case #2 - NISQ-Aware Circuit Transpilation

User Story: 2 - NISQ-Aware Circuit Transpilation	
User Story ID	US-#1
User Story Name	NISQ-Aware Circuit Transpilation
Actors	Quantum Algorithm Developer
Description	As a **Quantum Algorithm Developer**, I want to **provide a logical quantum circuit and a target quantum device**, so that I can **transpile the circuit to optimize it for the device's specific hardware topology and native gate set**.

Acceptance Criteria

Given I am a quantum developer, when I upload a 5-qubit circuit with non-adjacent CNOT gates and select 'ibmq_manila' as the target, then the system returns a transpiled circuit that uses only the native gates and has SWAP gates inserted to respect the qubit connectivity.

Scenarios

Normal Scenario	
Actor Action	**System Response**
User uploads a QuYAML file and specifies the target backend 'ibm_cairo'.	1. System parses the QuYAML file into a Qiskit object. 2. System retrieves the latest calibration data for 'ibm_cairo', including coupling map and native gates. 3. System performs a hardware-aware transpilation. 4. System decomposes all gates into the native basis gate set of the device. 5. System presents the optimized circuit and a comparison report to the user.

3.3 Use Case #3 - Quantum Support Vector Machine (QSVM) Design

User Story: 3 - QSVM Design	
User Story ID	US-#1
User Story Name	QSVM Design
Actors	QML Researcher
Description	As a **QML Researcher**, I want to **provide a classical dataset and a desired feature map**, so that I can **design and generate the full quantum circuit for a QSVM classifier**.

Acceptance Criteria

Given I am a QML researcher, when I upload a 2D binary classification dataset and specify a 'ZZFeatureMap', then the system generates a Python script that correctly sets up the QSVM in Qiskit, including data loading and feature mapping.

Scenarios

Normal Scenario	
Actor Action	**System Response**
User uploads a CSV file and prompts: ”Create a QSVM using a ZZFeatureMap for this data.”	1. System analyzes the CSV to determine the number of features. 2. The Planner Agent outlines the steps: Data Encoding, Feature Map Construction, QSVM Setup. 3. The Reasoner Agent generates the Qiskit code for each step. 4. The Verifier Agent confirms the number of qubits matches the number of classical features. 5. System presents the complete script and a diagram of the quantum kernel circuit.

3.4 Use Case #4 - Quantum Error Correction Code Debugging

User Story: 4 - QEC Code Debugging	
User Story ID	US-#1
User Story Name	QEC Code Debugging
Actors	Quantum Student
Description	As a **Quantum Student**, I want to **provide a faulty implementation of an error correction code**, so that I can **identify the logical error and suggest a correction**.

Acceptance Criteria

Given I am a student, when I provide a 3-qubit bit-flip code where a syndrome incorrectly maps to a correction gate, then the system identifies the incorrect logic and provides the corrected operation.

Scenarios

Normal Scenario	
Actor Action	**System Response**
User uploads a faulty QEC script and asks ”Where is the bug in my bit-flip code?”	1. System parses the code and recognizes the QEC structure. 2. The Reasoner Agent simulates the circuit for each possible single-qubit error. 3. It compares the resulting syndromes from its simulation with the user’s conditional logic. 4. It identifies a mismatch between the expected syndrome and the user’s corrective action. 5. System highlights the incorrect line, provides the fix, and explains the logic.

3.5 Use Case #5 - Portfolio Optimization with QAOA

	User Story: 5 - Portfolio Optimization with QAOA
User Story ID	US-#1
User Story Name	Portfolio Optimization with QAOA
Actors	Quantitative Analyst
Description	As a **Quantitative Analyst**, I want to **define a portfolio of financial assets**, so that I can **formulate the problem as a QUBO and generate the corresponding QAOA circuit**.

Acceptance Criteria

Given I am a quantitative analyst, when I provide data for 3 assets and a risk factor, then the system correctly formulates the QUBO, translates it into an Ising Hamiltonian, and generates a QAOA circuit in Qiskit.

Scenarios

Normal Scenario	
Actor Action	**System Response**
User provides financial data and the prompt: "Optimize this portfolio using QAOA."	1. The Planner Agent outlines steps: Formulate QUBO, Convert to Ising Hamiltonian, Construct QAOA Circuit. 2. The Reasoner Agent uses its mathematical capabilities to perform the QUBO formulation. 3. The Reasoner Agent then generates the Qiskit code for the QAOA ansatz. 4. The system presents the final script and a summary of the mathematical formulation.

3.6 Use Case #6 - Socratic Tutor for Grover's Algorithm

	User Story: 6 - Socratic Tutor for Grover's Algorithm
User Story ID	US-#1
User Story Name	Socratic Tutor for Grover's Algorithm
Actors	Student
Description	As a **Student**, I want to **ask for help understanding a complex quantum algorithm**, so that I can **receive guided, interactive instruction to build understanding**.

Acceptance Criteria

Given I am a student and I ask "How does Grover's algorithm work?", then the system responds not with the full answer, but with a guiding question to prompt my learning (e.g., "Great question! It starts with creating a superposition. Which gate does that?").

Scenarios

Normal Scenario	
Actor Action	**System Response**
User asks: "Help me build a Grover's search for the state $\|101\rangle$."	1. The agent adopts a "Tutor" persona and asks the user for the first step. 2. User correctly answers: "Apply Hadamard gates." 3. The agent affirms the answer and asks about the next component (the oracle). 4. The interactive process continues until the student has built the full circuit. 5. The agent provides the final code and a summary of concepts covered.

3.7 Use Case #7 - Heuristic-Based Algorithm Suggestion

User Story: 7 - Algorithm Suggestion	
User Story ID	US-#1
User Story Name	Algorithm Suggestion
Actors	Researcher
Description	As a **Researcher**, I want to **describe the mathematical structure of a problem**, so that I can **receive suggestions for suitable quantum algorithms**.

Acceptance Criteria

Given I am a researcher and I describe a problem that involves finding the minimum value of a combinatorial function, then the system identifies this as an optimization problem and suggests QAOA and VQE as primary candidates.

Scenarios

Normal Scenario	
Actor Action	**System Response**
User describes a problem related to finding the prime factors of a large number.	1. The Planner Agent identifies key terms: "factors", "large number", "period finding". 2. The Reasoner Agent accesses its knowledge base and connects these terms to Shor's algorithm. 3. It also identifies the Quantum Fourier Transform (QFT) as the core subroutine. 4. The Synthesizer Agent presents Shor's algorithm as the primary solution and explains its reliance on QFT.

3.8 Use Case #8 - Cross-Platform Performance Estimation

User Story: 8 - Performance Estimation	
User Story ID	US-#1
User Story Name	Performance Estimation
Actors	Quantum Developer
Description	As a **Quantum Developer**, I want to **provide a Qiskit circuit**, so that I can **receive an estimation of its performance on different real hardware backends**.

Acceptance Criteria

Given I am a developer and I submit a 4-qubit circuit, then the system returns a table comparing its estimated performance on 'ibm_cairo', 'ibmq_manila', and 'ibmq_kolkata' using their latest calibration data.

Scenarios

Normal Scenario	
Actor Action	**System Response**
User uploads a circuit and asks, "Which IBM backend is best for this circuit?"	1. The Planner Agent identifies the target devices and the user's circuit. 2. For each device, the Reasoner Agent retrieves the latest calibration data. 3. It performs a hardware-aware transpilation of the user's circuit for each device. 4. It constructs a noise model and runs a noisy simulation to estimate fidelity. 5. The Synthesizer Agent compiles the results into a comparison table and provides a recommendation.

3.9 Use Case #9 - Multi-Format Quantum Circuit Conversion

User Story: 9 - Circuit Conversion	
User Story ID	US-#1
User Story Name	Circuit Conversion
Actors	Researcher
Description	As a **Researcher**, I want to **provide a quantum circuit in a non-Qiskit format**, so that I can **convert it into both an executable Qiskit script and the QuYAML standard**.

Acceptance Criteria

Given I am a researcher and I paste a Python script using Google's Cirq library, then the system correctly identifies the gates and outputs a Qiskit script and a QuYAML file that produce the identical unitary operation.

Scenarios

Normal Scenario	
Actor Action	**System Response**
User pastes a snippet of Cirq code.	1. The Planner Agent identifies the source language as Cirq. 2. The Reasoner Agent iterates through the Cirq code, mapping each operation to its Qiskit equivalent. 3. Once the structure is understood, the Reasoner generates two outputs: the Qiskit script and the QuYAML definition. 4. The system presents both outputs to the user.

3.10 Use Case #10 - QGAN Analysis

User Story: 10 - QGAN Analysis	
User Story ID	US-#1
User Story Name	QGAN Analysis
Actors	QML Researcher
Description	As a **QML Researcher**, I want to **provide the results of a QGAN training run**, so that I can **receive an analysis of the training process and generated state quality**.

Acceptance Criteria

Given I am a QML researcher and I upload loss history from a QGAN that shows the discriminator loss going to zero, then the system correctly identifies this as potential mode collapse and explains the phenomenon.

Scenarios

Normal Scenario	
Actor Action	**System Response**
User uploads training data and the final state vector.	1. The Planner decides to first analyze the classical training data, then the quantum state. 2. The Reasoner Agent plots the loss curves and identifies patterns indicating mode collapse. 3. It then analyzes the final quantum state's properties (e.g., entropy) to confirm the diagnosis. 4. The Synthesizer Agent provides a full report, explaining the issue and suggesting next steps.

3.11 Use Case #11 - Interactive Quantum State Tomography

User Story: 11 - Interactive State Tomography	
User Story ID	US-#1
User Story Name	Interactive State Tomography
Actors	Experimental Physicist
Description	As a **Experimental Physicist**, I want to **provide a quantum circuit**, so that I can **receive a guided process to perform state tomography and interpret results**.

Acceptance Criteria

Given I am an experimental physicist and I provide a 2-qubit circuit, then the system correctly generates the 9 required measurement circuits, and after I provide outcome counts, it correctly computes and displays the 4x4 density matrix.

Scenarios

Normal Scenario	
Actor Action	**System Response**
User provides a circuit and requests state tomography.	1. The Planner Agent determines the required set of measurement bases (Pauli bases). 2. The Reasoner Agent generates the set of Qiskit circuits with appended basis change operations.
3. User runs these circuits and provides the results (counts).	4. The Reasoner Agent takes the counts, applies a reconstruction algorithm to compute the density matrix. 5. The Synthesizer Agent presents the density matrix, calculates fidelity and purity, and visualizes the result.

3.12 Use Case #12 - Molecular Ground State Energy Calculation

User Story: 12 - Molecular Ground State Energy Calculation	
User Story ID	US-#1
User Story Name	Molecular Ground State Energy Calculation
Actors	Computational Chemist
Description	As a **Computational Chemist**, I want to **input a molecule's standard identifier**, so that I can **receive its calculated ground state energy using a quantum algorithm**.

Acceptance Criteria

Given I am a computational chemist and I input ”H2” with a bond distance of 0.74 angstroms, then the system automates the VQE process and returns a ground state energy value accurate to within a small tolerance of the known FCI value.

Scenarios

Normal Scenario	
Actor Action	**System Response**
User inputs ”Calculate the ground state energy of H2 at 0.74 angstroms.”	1. The Planner recognizes this as a specific instance of Use Case #1. 2. The Reasoner uses its knowledge base to select appropriate defaults (basis, optimizer, ansatz). 3. It performs the entire VQE calculation on a simulator. 4. The Synthesizer Agent presents the final energy, a convergence plot, and the QuYAML of the final optimized ansatz.

4 Non-Functional Requirements

NFR-1: Performance The system shall return a complete response for a standard 3-qubit Grover's algorithm query in under 30 seconds.

NFR-2: Usability The user interface shall be clean, intuitive, and require no training for a user familiar with quantum computing concepts.

NFR-3: Reliability The system's core functionalities shall have an uptime of 99% during the demo period.

NFR-4: Token Efficiency The system shall use the QuYAML format for circuit definition, which must be at least **60% more token-efficient** than an equivalent JSON representation to minimize API costs and latency.

Tools

Category	Component / Tool	Role in QThink
AI Core / Reasoning	**K2 Think API**	Provides state-of-the-art *mathematical reasoning* essential for accurate quantum simulation and optimization.
Context & Knowledge	**Context-Aided Generation (CAG) Store**	A pre-cached vector store containing Qiskit documentation and key textbook excerpts for low-latency, grounded responses.
Backend & Orchestration	**FastAPI**	High-performance Python framework used for the **Backend Orchestrator** to manage the workflow.
Agent Framework	**Lang Chain**	Used to implement the *agentic backend* for intent classification, tool-use, and multi-step reasoning.
Frontend	**React.js**	Used to build the clean, intuitive, and highly usable web interface (NFR-2).
Data Interchange	**QuYAML**	A novel, domain-specific YAML schema used for circuit definition, achieving $>$ 60% **token efficiency** (NFR-4).
Parsing	**PyYAML**	Python library used by the Input Parser to handle the custom QuYAML schema.
External Verification	**IBM Quantum API**	Used by the **Verifier Agent** to execute Qiskit code on a public simulator for real-world result verification.
Logging / Persistence	**SQLite Database**	Simple local database used to log user interactions, outputs, and errors for the **Darwin Gödel Machine** loop (FR-7).
Visualization (Proposed)	**Qwen-VL**	Proposed integration to enhance the visualization of quantum state vectors and circuit diagrams.

Table 1: Software Components and Their Roles in QThink