Software Requirements Specification (SRS) Document

Stock Market Prediction Using ML, DL, QML, and QNN

1. Introduction

1.1 Purpose

The purpose of this document is to define the requirements for a **Stock Market Prediction System** that utilizes Machine Learning (ML), Deep Learning (DL), Quantum Machine Learning (QML), and Quantum Neural Networks (QNN). The system aims to predict stock price trends using a combination of traditional and quantum-based models, improving accuracy and decision-making for investors and traders.

1.2 Scope

This system will provide insights into stock price movements based on historical data and sentiment analysis. It will:

- Fetch live and historical stock market data.
- Process and clean the data for feature engineering.
- Train predictive models (ML, DL, QML, QNN) on stock price trends.
- Provide visualized stock market predictions and performance comparisons.
- Offer a web-based interface for users to interact with predictions and insights.

1.3 Definitions, Acronyms, and Abbreviations

- **ML (Machine Learning)**: Algorithms that enable computers to learn from data patterns.
- **DL (Deep Learning)**: A subset of ML using neural networks with multiple layers.
- **QML (Quantum Machine Learning)**: Application of quantum computing principles to ML.
- **QNN (Quantum Neural Networks)**: Neural networks utilizing quantum circuits.
- **SMA (Simple Moving Average)**: A stock market indicator to smooth price trends.
- **RSI (Relative Strength Index)**: A technical analysis indicator for market trends.

1.4 References

- Stock Market Data API: Yahoo Finance (yFinance)
- Quantum Computing: PennyLane Documentation
- Flask Web Framework: Flask Documentation

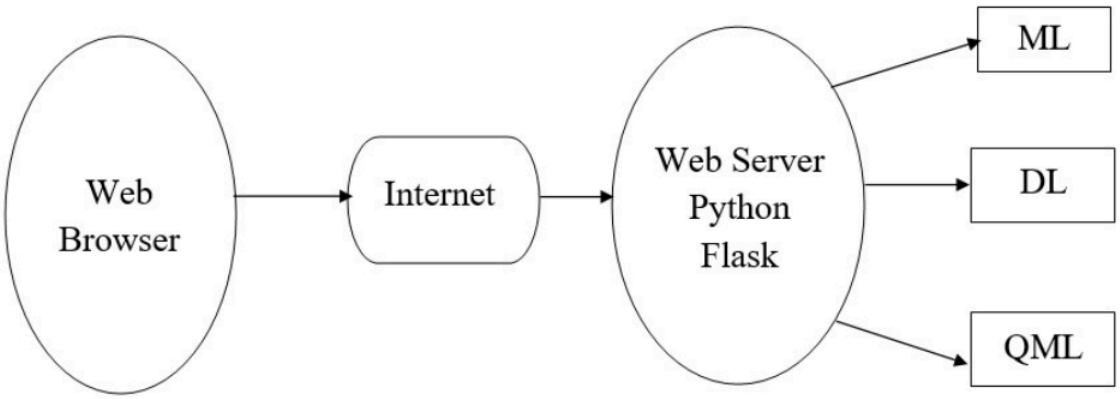
2. Overall Description

2.1 Product Perspective

This system will be an AI-powered stock market forecasting tool that combines multiple predictive models. It will have both a **backend** (Python, Flask) and a **frontend** (HTML, CSS, JavaScript) for visualization and user interaction.

2.2 Product Functions

- Fetch real-time and historical stock data.
- Perform feature engineering and data preprocessing.
- Train and evaluate ML, DL, QML, and QNN models.
- Provide predictions and accuracy comparisons.
- Display data through an intuitive user interface.



2.3 User Characteristics

- **Traders & Investors**: Use predictions for decision-making.
- **Researchers & Students**: Explore the impact of quantum computing in finance.
- **Developers**: Modify and enhance the system.

2.4 Constraints

- Requires stable internet for live stock data fetching.
- Computation-intensive for quantum models.
- Limited dataset availability for QML and QNN training.

3. Specific Requirements

3.1 Functional Requirements

1. **Stock Data Collection**
    - Fetch historical data from Yahoo Finance.
    - Support multiple stocks for analysis.
2. **Data Processing & Feature Engineering**
    - Calculate SMA, RSI, and other indicators.
    - Handle missing values and outliers.
3. **Model Training & Prediction**
    - Train ML, DL, QML, and QNN models.
    - Evaluate models based on accuracy metrics.
4. **User Interface & Visualization**
    - Display stock charts, trends, and predictions.
    - Show model performance comparisons.
5. **Flask Backend API**
    - Handle requests for stock data and predictions.
    - Manage user interactions with the models.

3.2 Non-functional Requirements

- **Performance**: Predictions should be generated in under 5 seconds.
- **Scalability**: Should handle multiple stock requests.
- **Security**: Secure API endpoints for data access.
- **Usability**: UI should be responsive and user-friendly.

4. External Interface Requirements

4.1 User Interfaces

- **Web Dashboard**: Displays stock data and predictions.
- **Input Fields**: Users select stocks and date ranges.
- **Graphs & Charts**: Data visualization for trends.

4.2 Hardware Interfaces

- **System Requirements**: Minimum 8GB RAM, GPU recommended.
- **Quantum Computing Support**: Compatible with PennyLane for QML & QNN.

4.3 Software Interfaces

- **Python Libraries**: scikit-learn, TensorFlow, PennyLane, Flask.
- **Frontend Technologies**: HTML, CSS, JavaScript, Chart.js for graphs.

4.4 Communication Interfaces

- **API Endpoints**: Flask-based backend API.
- **Database**: SQLite/PostgreSQL for storing stock data.

5. Other Requirements

- **Testing Strategy**: Unit tests for model accuracy validation.
- **Deployment**: Dockerized for cloud deployment.

6. Appendices

- **Future Enhancements**: Expand to cryptocurrency prediction.
- **References**: Quantum computing research papers, financial datasets.