1. Overview

This project implements a simple single-node key-value data store using **Apache Cassandra** as the backend and a **non-blocking I/O (NIO)** communication layer to handle client-server interactions. The goal is to extend the provided template code (Client.java and SingleServer.java) to create functional subclasses MyDBClient and MyDBSingleServer that exchange requests and responses asynchronously while executing queries on a local Cassandra instance.

2. Design Goals

- Maintain **non-blocking request/response** behavior.
- Ensure **one-to-one mapping** between requests and responses using unique identifiers.
- Connect to an existing **Cassandra keyspace** and execute arbitrary CQL queries.
- Pass all provided JUnit tests (GraderSingleServer.java).

3. Architecture

The design follows a **client–server model** over NIO sockets:

1. **MyDBClient**
    - Extends Client.java.
    - Sends query strings to the server using the inherited non-blocking send() method.
    - Each outgoing request is assigned a **unique request ID** (UUID).
    - Stores callbacks in a local map keyed by request ID.
    - When a response arrives via handleResponse(), it parses the response, extracts the ID, and invokes the corresponding callback.
    - Implements callbackSend() to ensure callbacks trigger **only after the matching response** is received.

2. **MyDBSingleServer**
    - Extends SingleServer.java.
    - Upon receiving a request, it examines whether a UUID is present and **removes it prior to query execution** to ensure valid CQL syntax
    - Next, it extracts the CQL query string and executes it on the **local Cassandra instance** via the DataStax Java driver.
    - Wraps the query result (or error message) into a response string prefixed with the same request ID.
    - Sends the response asynchronously to the client.
    - Manages Cassandra resources by connecting to the provided **keyspace (‘demo’)** and closing the Cluster and Session cleanly in close().
