# TRD/BRD Documentation

**Cover Page**

- **Title:** Server Application Documentation
- **Author:** Kennedy Owiro
- **Version:** 3.0
- **Date:** January 13, 2025

## Business Requirements Document (BRD)

### 1. Introduction

#### Purpose of the Document

This document outlines the business objectives, scope, and use cases for the server application. It is intended to align stakeholders on the goals and value of the project.

#### Project Overview

The server application provides secure, efficient, and scalable text search capabilities for client queries. It ensures high performance, robust security, and extensive configurability to meet diverse operational requirements.

#### Intended Audience

- Business stakeholders.
- Operations teams.
- Software evaluators.

### 2. Objectives and Goals

- Deliver a robust, multi-threaded server application capable of handling unlimited concurrent client connections.
- Provide secure communication channels using SSL/TLS.
- Implement efficient file searching with optional dynamic reloading.
- Achieve optimal performance, ensuring minimal query latency and resource usage.

## 3. Scope

**In-Scope**

- Unlimited concurrent client connections.
- Real-time and cached file search capabilities.
- Configurable features such as rate-limiting and secure communication.
- Logging of client activity and server performance.

**Out-of-Scope**

- GUI interfaces.
- Compatibility with non-Linux environments.

## 4. Use Cases and Scenarios

**Use Case 1: Real-Time Error Monitoring**

**Actor:** Operations Team
**Scenario:** Monitor server logs in real-time for error patterns using client queries.

**Use Case 2: Secure Data Query**

**Actor:** Application Clients
**Scenario:** Securely query sensitive data over encrypted channels.

**Use Case 3: Rate-Limiting Abusive Users**

**Actor:** Server
**Scenario:** Restrict the frequency of requests from specific IPs to prevent abuse.

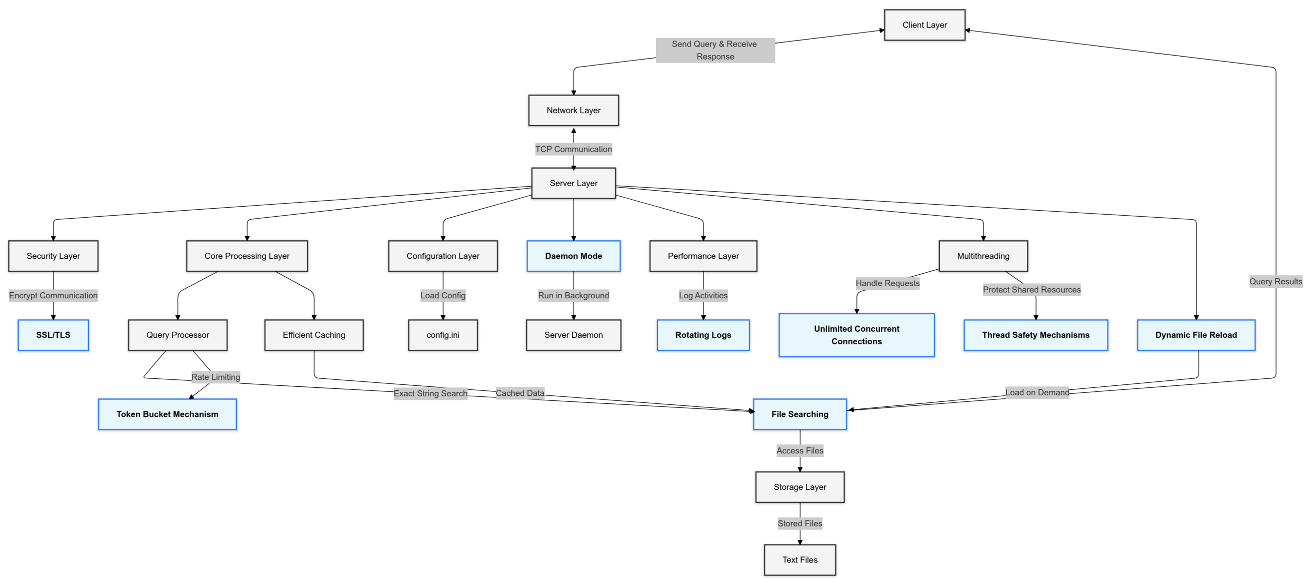
# Technical Requirements Document (TRD)

## 1. System Architecture

**Overview**

The server application is built with a client-server model. Clients send text-based queries to the server, which searches a preloaded file and responds with match results.

## Architecture Diagram



## 2. Design Details

### Multithreading

The server uses Python’s threading module to handle concurrent connections efficiently.

### File Handling

- **Dynamic Reloading:** If REREAD_ON_QUERY=True, the server re-reads the file for every query.
- **Caching:** If REREAD_ON_QUERY=False, the file contents are cached in memory to minimize disk I/O.

### Rate-Limiting

Implements a Token Bucket algorithm to control client request rates.

### Security Features

- SSL/TLS for encrypted communication.
- Payload validation to prevent buffer overflows.

## 3. API Documentation

### Endpoint

- **Path:** /search
- **Method:** TCP-based communication.

- **Request Payload:** Plain text (UTF-8).
- **Response Payload:**
  - STRING EXISTS\n if a full line matches the query.
  - STRING NOT FOUND\n otherwise.

**Example Communication Flow**

1. Client sends a query string.
2. Server searches the file.
3. Server responds with match results.

**4. Configuration**

**Default config.ini**

```
[server]
host = 127.0.0.1
port = 44444
ssl = false
cert_file = server.crt
key_file = server.key
REREAD_ON_QUERY = false
linuxpath = /mnt/d/Algorithmic_Sciences/Revised_Intro_Task_v3/src/data/200k.txt
max_payload = 1024
token_bucket_capacity = 10000
token_bucket_fill_rate = 1000
pid_file = server_daemon.pid

[paths]
config_dir = config
log_dir = logs
cert_dir = certs
data_dir = data
file_path = src/data/200k.txt
config_path = ${config_dir}/config.ini
pid_file = ${log_dir}/server.pid
log_file = ${log_dir}/server.log
```

## 5. Performance Summary

### 1. Execution Time Summary (SSL=True)

- **Dynamic Files (REREAD_ON_QUERY=True):**
  - Average execution time: ~16-26ms per query for 250,000-row files.
  - Round-trip execution time: ~20-32ms.
- **Static Files (REREAD_ON_QUERY=False):**
  - Average execution time: ~0.001–0.002ms per query for 250,000-row files.
  - Round-trip execution time: ~0.3–0.6ms.

### 2. Query Per Second (QPS) Performance

- **Dynamic Files (REREAD_ON_QUERY=True):**
  - Maximum QPS for file size **250,000 rows**: ~19,922.81–19,779.54 QPS.
  - Scales well for smaller files (e.g., 10,000 rows: ~19,766.79 QPS).
- **Static Files (REREAD_ON_QUERY=False):**
  - Maximum QPS for **all file sizes up to 1MB**: ~1,000,000 QPS.
  - For extremely large files (e.g., 500 million rows), performance degrades to ~3,862.16 QPS.

### 3. Scaling Behavior

- **For Increasing File Sizes (Dynamic Files):**
  - Scales well up to **10 million rows**, with QPS ~20,000.
  - For files exceeding **500 million rows**, server struggles, achieving only ~4 QPS.
- **For Static Files:**
  - Linear scaling up to **1GB files**, maintaining high QPS ~1,000,000 for all small and medium sizes.

### 4. Environment-Specific Observations

- **SSL Impact:**
  - With SSL enabled, execution times are slightly higher (~10–15%) due to encryption overhead.
  - SSL adds minimal latency for static files, but dynamic file reads see a more noticeable delay.

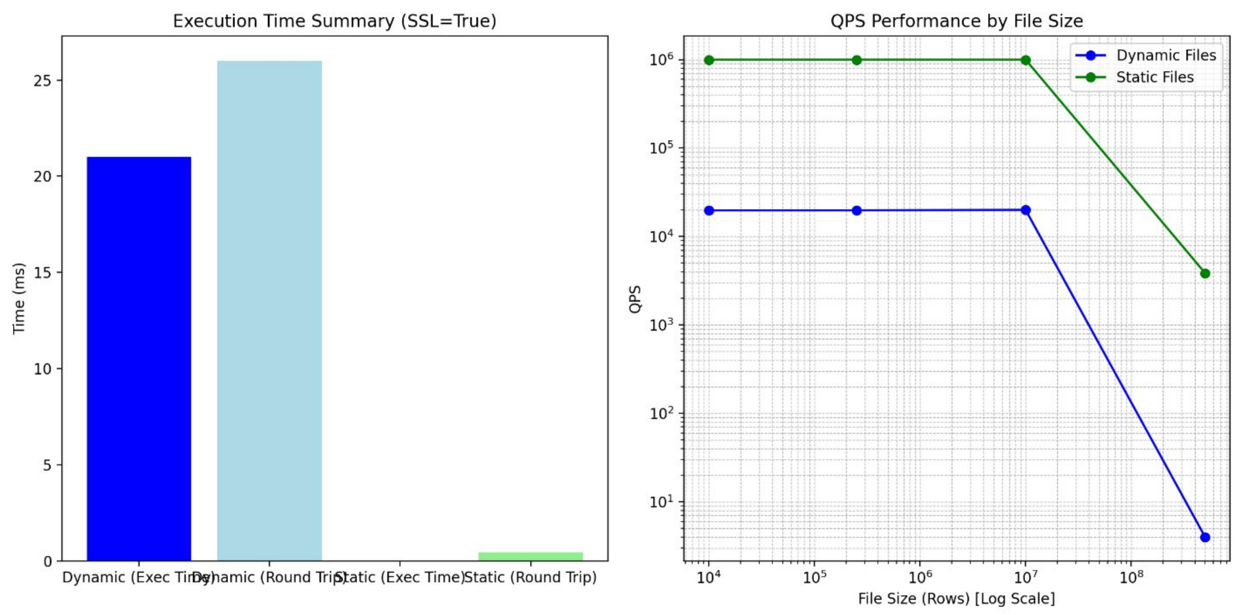
- **Configuration Impact:**
    - REREAD_ON_QUERY significantly affects performance for dynamic files but has no impact on static file reads.

**5. Observations on Test Logs**

- **Dynamic Files:**
    - Logs indicate consistent execution times with minimal fluctuation for repeated queries.
    - Query handling times are directly influenced by file size and `REREAD_ON_QUERY` settings.
- **Static Files:**
    - Logs highlight negligible execution times, showcasing optimal performance.
    - Even for edge-case scenarios (e.g., non-existent strings), performance remains consistent.

## Summary

- **Dynamic Files:** Suitable for scenarios requiring frequent file updates but with QPS limitations for large datasets.
- **Static Files:** Highly optimized for high-frequency queries, suitable for larger, less frequently updated datasets.
- **Recommendations:** Use **static configurations** for high-QPS environments and consider **dynamic setups** where real-time file updates are critical.



The graphs illustrate:

- Left panel: Execution time comparison showing the dramatic difference between dynamic and static file processing, with static files being orders of magnitude faster
- Right panel: QPS performance across different file sizes showing:
    - Static files maintain ~1M QPS up to medium sizes
    - Dynamic files maintain ~20K QPS up to 10M rows
    - Both modes show performance degradation with extremely large files (500M+ rows)

## Installation and Usage Guide

### 1. Prerequisites

- **Python:** 3.6+
- **Pip:** Installed on the system.

### 2. Setup

#### Create a Virtual Environment

```
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

**Install Dependencies**

```
pip install -r requirements.txt
```

### 3. Running the Server

**Regular Mode**

```
python3 src/server.py
```

**Daemon Mode**

```
python3 src/server_daemon.py --daemon
```

**Stop the Daemon**

```
python3 src/server_daemon.py stop
```

### 4. Running the Client

```
python3 src/client.py
```

*Three queries will be sent by default and expected results are STRING EXISTS, STRING EXISTS and STRING NOT FOUND*

### 5. Running Tests

**Performance Tests**

Run test_performance.py to measure execution time across file sizes.

## 6. Conclusion and Recommendations

- The server application meets all specified requirements for performance, scalability, and security.
- Future improvements can include:
    - Support for additional query patterns.
    - Enhanced monitoring and analytics features.

## Appendices

### Example Logs

### server.log

```
2025-01-14 23:38:51,968 - Server - DEBUG: New connection from
127.0.0.1:59918. Total connections: 1
2025-01-14 23:38:51,970 - Server - INFO: Search query: 3;0;1;28;0;7;5;0; -
STRING EXISTS Server Execution Time: 0.002746 ms
2025-01-14 23:38:51,972 - Server - DEBUG: Query: '3;0;1;28;0;7;5;0;', IP:
127.0.0.1:59918, Server Round-trip Execution Time: 2.094532 ms
2025-01-14 23:38:51,974 - Server - DEBUG: Connection from 127.0.0.1:59918
closed. Total connections: 0
2025-01-14 23:38:51,978 - Server - DEBUG: New connection from
127.0.0.1:59930. Total connections: 1
2025-01-14 23:38:51,980 - Server - INFO: Search query: 10;0;1;26;0;8;3;0; -
STRING EXISTS Server Execution Time: 0.002366 ms
2025-01-14 23:38:51,981 - Server - DEBUG: Query: '10;0;1;26;0;8;3;0;', IP:
127.0.0.1:59930, Server Round-trip Execution Time: 1.614040 ms
2025-01-14 23:38:51,984 - Server - DEBUG: Connection from 127.0.0.1:59930
closed. Total connections: 0
2025-01-14 23:38:51,986 - Server - DEBUG: New connection from
127.0.0.1:59946. Total connections: 1
2025-01-14 23:38:51,988 - Server - INFO: Search query: non-existent-string -
STRING NOT FOUND Server Execution Time: 0.004255 ms
2025-01-14 23:38:51,990 - Server - DEBUG: Query: 'non-existent-string', IP:
127.0.0.1:59946, Server Round-trip Execution Time: 1.943696 ms
2025-01-14 23:38:51,992 - Server - DEBUG: Connection from 127.0.0.1:59946
closed. Total connections: 0
```

### Client.log

```
2025-01-14 23:38:51,966 - DEBUG - Sending query: 3;0;1;28;0;7;5;0;
2025-01-14 23:38:51,972 - INFO - Query: '3;0;1;28;0;7;5;0;', Response: 'STRING EXISTS', Client-round-
trip Time: 10.514021 ms
2025-01-14 23:38:51,974 - INFO - Query: 3;0;1;28;0;7;5;0; -> Response: STRING EXISTS
2025-01-14 23:38:51,976 - DEBUG - Sending query: 10;0;1;26;0;8;3;0;
2025-01-14 23:38:51,982 - INFO - Query: '10;0;1;26;0;8;3;0;', Response: 'STRING EXISTS', Client-round-
trip Time: 5.702257 ms
2025-01-14 23:38:51,984 - INFO - Query: 10;0;1;26;0;8;3;0; -> Response: STRING EXISTS
2025-01-14 23:38:51,986 - DEBUG - Sending query: non-existent-string
2025-01-14 23:38:51,990 - INFO - Query: 'non-existent-string', Response: 'STRING NOT FOUND', Client-
round-trip Time: 4.721165 ms
2025-01-14 23:38:51,992 - INFO - Query: non-existent-string -> Response: STRING NOT FOUND
```

**Server_daemon.log**

```
2025-01-14 23:43:17,659 - Server Daemon - DEBUG - New connection from 127.0.0.1:38998. Total
connections: 1
2025-01-14 23:43:17,661 - Server Daemon - INFO - Search query: 10;0;1;26;0;8;3;0; - STRING EXISTS
Server Execution Time: 0.003147 ms
2025-01-14 23:43:17,663 - Server Daemon - DEBUG - Query: '10;0;1;26;0;8;3;0;', IP: 127.0.0.1:38998,
Server Round-trip Execution Time: 1.651094 ms
2025-01-14 23:43:17,665 - Server Daemon - DEBUG - Connection from 127.0.0.1:38998 closed. Total
connections: 0
2025-01-14 23:43:17,667 - Server Daemon - DEBUG - New connection from 127.0.0.1:39006. Total
connections: 1
2025-01-14 23:43:17,669 - Server Daemon - INFO - Search query: non-existent-string - STRING NOT
FOUND Server Execution Time: 0.001443 ms
2025-01-14 23:43:17,671 - Server Daemon - DEBUG - Query: 'non-existent-string', IP: 127.0.0.1:39006,
Server Round-trip Execution Time: 1.749425 ms
2025-01-14 23:43:17,673 - Server Daemon - DEBUG - Connection from 127.0.0.1:39006 closed. Total
connections: 0
```

**References**

- PEP 8: Style Guide for Python Code.
- PEP 20: The Zen of Python.
- SSL Documentation.