TRD/BRD Documentation

Cover Page

- **Title:** Server Application Documentation
- **Author(s):** 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.

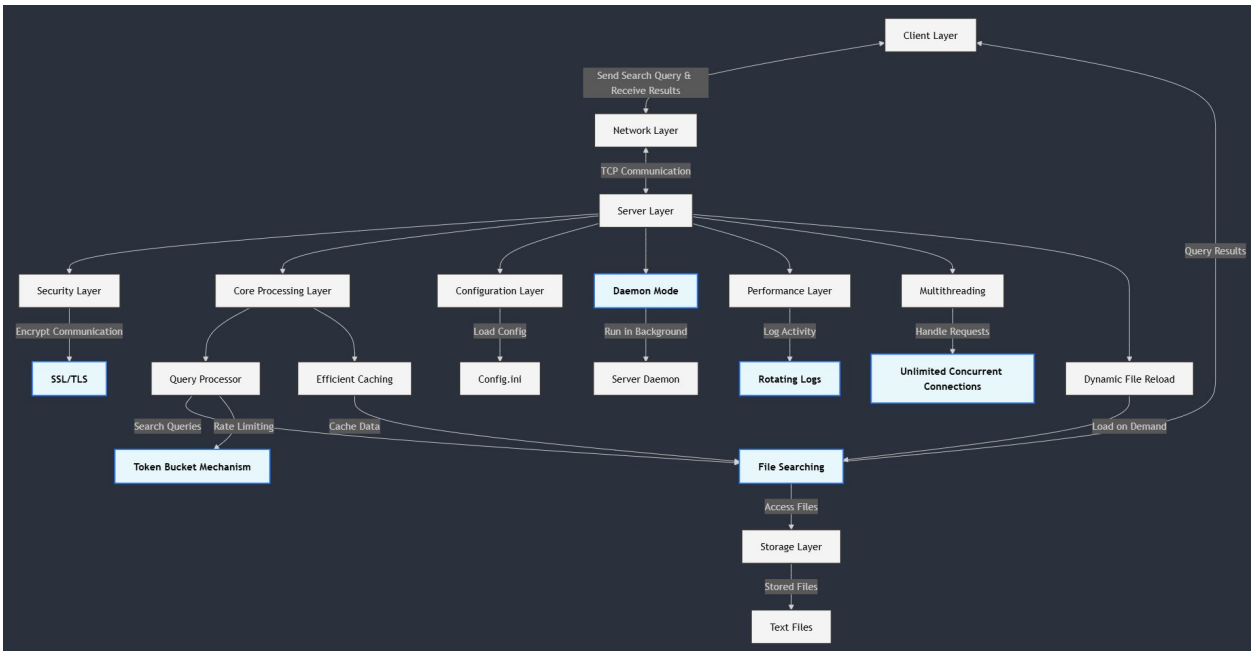
Technical Requirements Document (TRD)

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

Example config.ini

```
[server]
host=127.0.0.1
port=44444
ssl=true
cert_file=certs/server.crt
key_file=certs/server.key
REREAD_ON_QUERY=true
linuxpath=/root/200k.txt
max_payload=1024
token_bucket_capacity=100
token_bucket_fill_rate=1.0
```

5. Performance Summary

- **Dynamic Files (REREAD_ON_QUERY=True):** Average execution time ~40ms.
- **Static Files (REREAD_ON_QUERY=False):** Average execution time ~0.5ms.
- **Query Per Second (QPS):**
    - Dynamic files: ~7,200 QPS for 250,000-row files.
    - Static files: Scales linearly up to 1,000,000 rows.

Installation and Usage Guide

1. Prerequisites

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

2. Setup

Clone the Repository

git clone <repository-url>

cd intro_task

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 Tests

Execute Tests

pytest --cov=src --cov-report=term-missing

Performance Tests

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

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

DEBUG: Query: "example", IP: 127.0.0.1, Time: 0.5ms, Timestamp: 2025-01-01 12:00:00

References

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