MBARARA UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF COMPUTING AND INFORMATICS

COURSE UNIT: SOFTWARE ENGINEERING INDUSTRIAL MINI PROJECT II

COURSE CODE: SWE4106

Academic Year: 2024/2025

Semester: One

Student name: **BWESIGYE TREASURE**

Regno: **2021/BSE/145/PS**

Student number: **2100603048**

Software Requirements Document (SRD) for Portable Matrix Library

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to outline the requirements for the Portable Matrix Library. This library provides developers with a suite of optimized tools specifically designed for:

Matrix multiplication (utilized in graphics, machine learning, and engineering simulations).

Fourier transforms (primarily used in signal processing and image analysis).

Optimization techniques, such as gradient descent (for artificial intelligence and various computational problems).

The library aims to reduce complexity, improve performance, and enhance portability across different systems. [4][5]

1.2 Scope

This library can be integrated into various applications, including, but not limited to:

Machine learning frameworks (**e.g., TensorFlow, PyTorch**). [1]

Signal and image processing applications.

Scientific computing systems requiring matrix operations and optimization routines. [2][3]

2. Overall Description

2.1 Product Perspective

The Portable Matrix Library is an independent library that can be integrated into existing software solutions or standalone applications. It serves as a provider for essential mathematical operations [2] that facilitate advanced data processing.

2.2 Product Functions

Matrix Operations: Support for basic (addition, subtraction) and advanced (multiplication, inversion) matrix operations. [2]

Fourier Transforms: Implementation of Fast Fourier Transform (FFT) algorithms for efficient data processing.

Optimization Routines: Includes gradient descent and other optimization techniques tailored for machine learning models.

2.3 User Classes and Characteristics

Machine Learning Developers: Familiar with mathematical operations and optimization techniques. Requires efficient algorithms for training large models. [1]

Signal Processing Engineers: Need reliable and fast methods for signal transformation and data analysis.

Game Developers: Require high-performance matrix calculations for real-time rendering and transformations.

2.4 Operating Environment

The library will support the following environments:

Operating Systems: Linux and Windows.

Programming Languages: C, C++, Python, Java and Rust.

2.5 Constraints

The library must ensure compatibility with various compilers for each supported language.

Cross-platform support may require ***conditional compilation or specific bindings.***

3. Requirements

3.1 Functional Requirements

1. Matrix Multiplication: Capable of performing matrix multiplication for input matrices of size up to n x n efficiently.

2. Fourier Transforms: Must implement **FFT** (***Fast Fourier Transformations***) for rapid Fourier transformation processes on input data arrays.

3. Optimization Routines: Support for gradient descent, including options for initialization, learning rates, and iteration limits.

Using ***C programming language*** as it is a native language that provides ease with portability into other languages.

3.2 Performance Requirements

1. Computational Efficiency: The library should minimize execution time for all operations, utilizing efficient algorithms and data structures.

2. Memory Management: Ensure minimal overhead, particularly in language bindings for Python and Rust, to avoid performance slowdowns.

3.3 Portability Requirements

1. **Integration Support:** Provide seamless integration capabilities into ***C, Java, C++, Python, and Rust.***

2. **Cross-platform Compatibility:** Ensure functionality on various operating systems ***(Linux, Windows)*** without significant modifications.

4. Use Cases

4.1 Machine Learning Developers

Scenario: Developers will utilize the gradient descent function for optimizing machine learning model parameters, enhancing model accuracy and performance.

4.2 Signal Processing Engineers

Scenario: Engineers will apply Fourier transforms to analyze and process signal data quickly, ensuring accurate transformations for real-time applications.

4.3 Game Developers

Scenario: Developers will employ matrix multiplication functions for rendering transformations in 3D graphics, contributing to efficient rendering pipelines in games.

5. User Interface Requirements

5.1 Command-Line Interface (CLI)

Provides a simple CLI for users to execute library functions directly for testing purposes.

5.2 Application Programming Interface (API)

Documentation for using the library in various languages, including example code snippets and function descriptions.

6. Non-Functional Requirements

6.1 Usability

The library should provide clear documentation, setup instructions, and usage examples for quick and easy integration.

6.2 Security

Ensure that input validation is performed to protect against common vulnerabilities such as buffer overflows.

6.3 Reliability

The library should pass all unit tests and provide consistent results across different operations and platforms.

6.4 Scalability

The library must be capable of handling increasingly large matrices and data sets without significant degradation in performance.

7. Testing Requirements

Unit Testing: Develop unit tests for each library function to ensure correctness.

Performance Testing: Benchmark library operations under various scenarios to evaluate execution time and memory usage.

Cross-platform Testing: Validate functionality and performance on all supported operating systems.

8. Maintenance and Support Requirements

Provide regular updates for bug fixes and optimizations based on user feedback.

Offer user support through issue tracking on platforms like ***GitHub*** **[6]** with documentation.

9. Assumptions and Dependencies

Assumes a stable development environment with access to ***compiler tools for C, C++, Python, and Rust.***

Dependencies may include mathematical libraries **(like BLAS or LAPACK)** for optimized operations.

10. Appendices

References.

[1] S. Russell and P. Norvig, “Artificial Intelligence: A Modern Approach”, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

[2] I. M. Cohen, “Computer-Aided Design of Electronic Circuits”, 2nd ed. New York, NY, USA: McGraw-Hill, 2015.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, “Introduction to Algorithms”, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[4] D. C. Li and I. J. H. Tran, “Creating Software Libraries: A Methodological Approach”, 1st ed. London, UK: Springer, 2018.

[5] D. W. Karban, “Writing Effective Software Documentation”, 2nd ed. New York, NY, USA: Addison-Wesley, 2015.

[6] **https://github.com/treasure16522/Portable-library.git** ***Accessed on 12th December, 2024.***