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**

Table of Contents

Software Design Document

1. Introduction

1.1 Purpose

The purpose of this document is to provide a comprehensive overview of the design and architecture of the Portable Matrix Library. This library was developed to address computational needs in CAD, algorithms, and machine learning by integrating matrix multiplication, FFT, and gradient descent functionalities into a single, portable tool.

1.2 User Classes and Characteristics

- **Machine Learning Developers:** Familiar with mathematical operations and optimization techniques, requiring efficient algorithms for training large models.
- **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. System Overview

The Portable Matrix Library was designed to simplify complex computations, combining multiple functionalities into one tool. It is optimized for performance and usability across diverse fields.[1]

3. Design Considerations

3.1 Goals

Provide fast and reliable matrix operations.

Ensure cross-platform compatibility.

Minimize computational resource consumption.

3.2 Constraints

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

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

4. System Architecture

4.1 High-Level Architecture

The library is modular, with separate components for:

- **Matrix Operations**: Handles multiplication and related functions.

- **FFT Module:** Optimized for signal processing tasks.

- **Gradient Descent Module:** Supports machine learning optimizations.

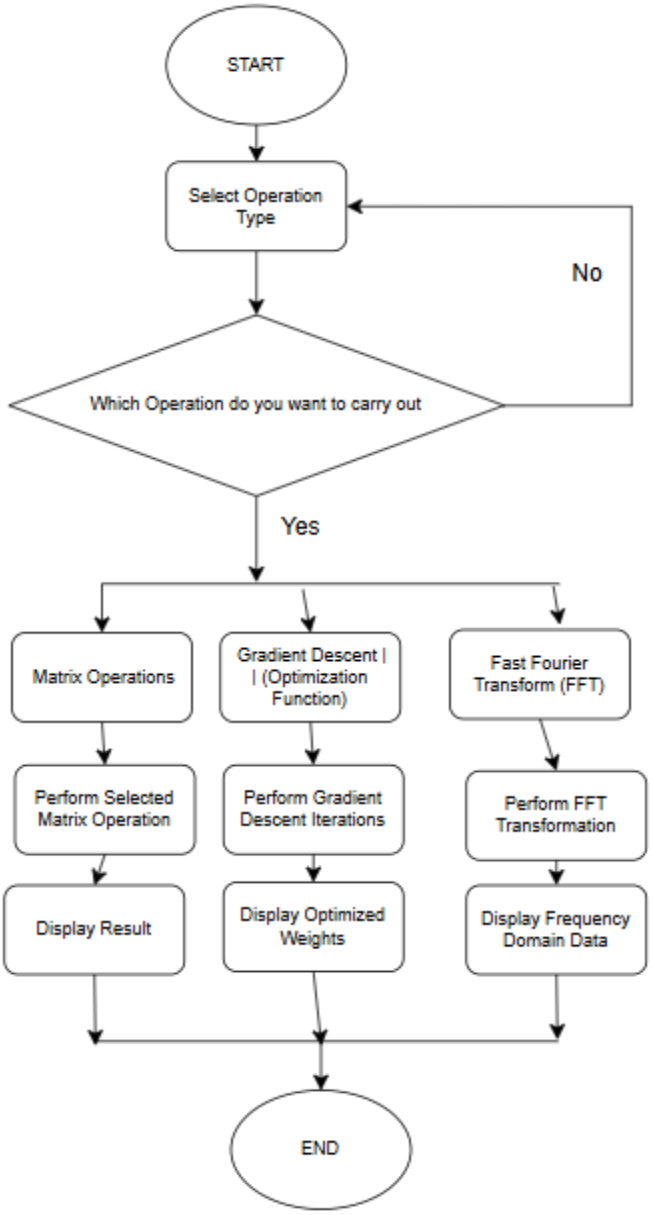
The library will support:

Operating Systems : Linux and Windows.

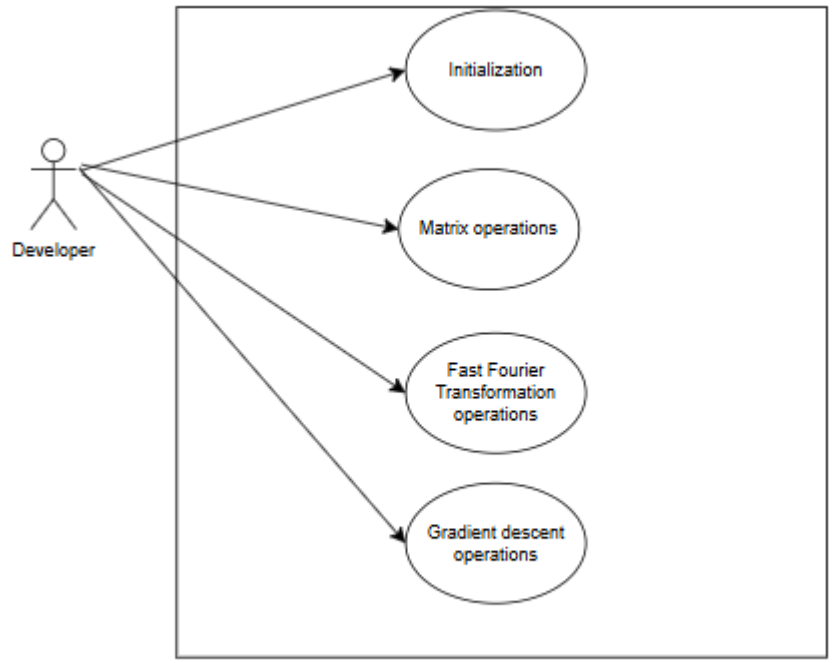
Programming Languages : C, C++, Python, Java, and Rust.

4.2 System Design Diagrams

Flow chart



Use case diagram.



5. Detailed Design

5.1 Matrix Multiplication

Algorithm: SUMMA (Scalable Universal Matrix Multiplication Algorithm) was implemented to achieve high scalability and efficiency.

Input/Output: Accepts two matrices and outputs their product.

5.2 FFT Module

Algorithm: Implements Cooley-Tukey FFT for optimized performance in signal processing.

Input/Output: Processes arrays to transform data between time and frequency domains.

5.3 Gradient Descent

Algorithm: Supports batch, stochastic, and mini-batch gradient descent with adjustable learning rates.

Input/Output: Takes feature matrices and target values to calculate optimized parameters.

6. User Interface Design

6.1 Command-Line Interface (CLI)

The library provides a CLI for:

Executing matrix operations.

Running FFT.

Optimizing machine learning models.

7. References

1. Badiru, A. B. (2008). Project Management Essentials. CRC Press, pp. 95-118.