The National Institute of Engineering, Mysuru
Department of Computer Science & Engineering (AIML)

GLACIER.ML

MINOR PROJECT EVALUATION - PHASE 1

SEPTEMBER 10, 2025
Batch - F11

Project guide: Ms. Mahe Mubeen Aktar D
Team: Skandan C.Y (4NI23CI103)

TABLE OF CONTENTS

01 Introduction

02 Literature Review

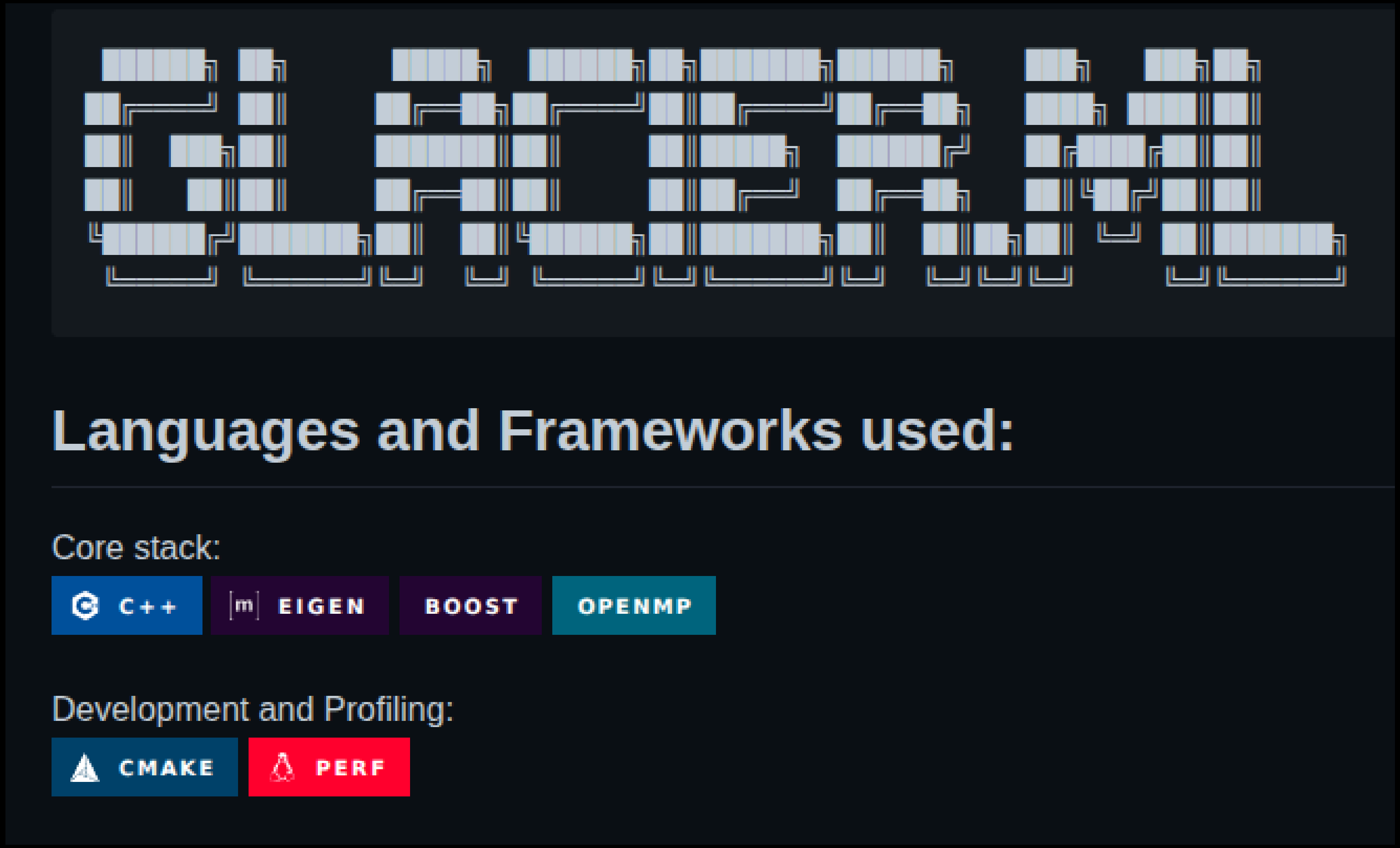
03 Existing system and formulation of proposed system

04 System design

05 System requirements

06 Conclusion

07 References



Languages and Frameworks used:

Core stack:

C++ EIGEN BOOST OPENMP

Development and Profiling:

CMAKE PERF

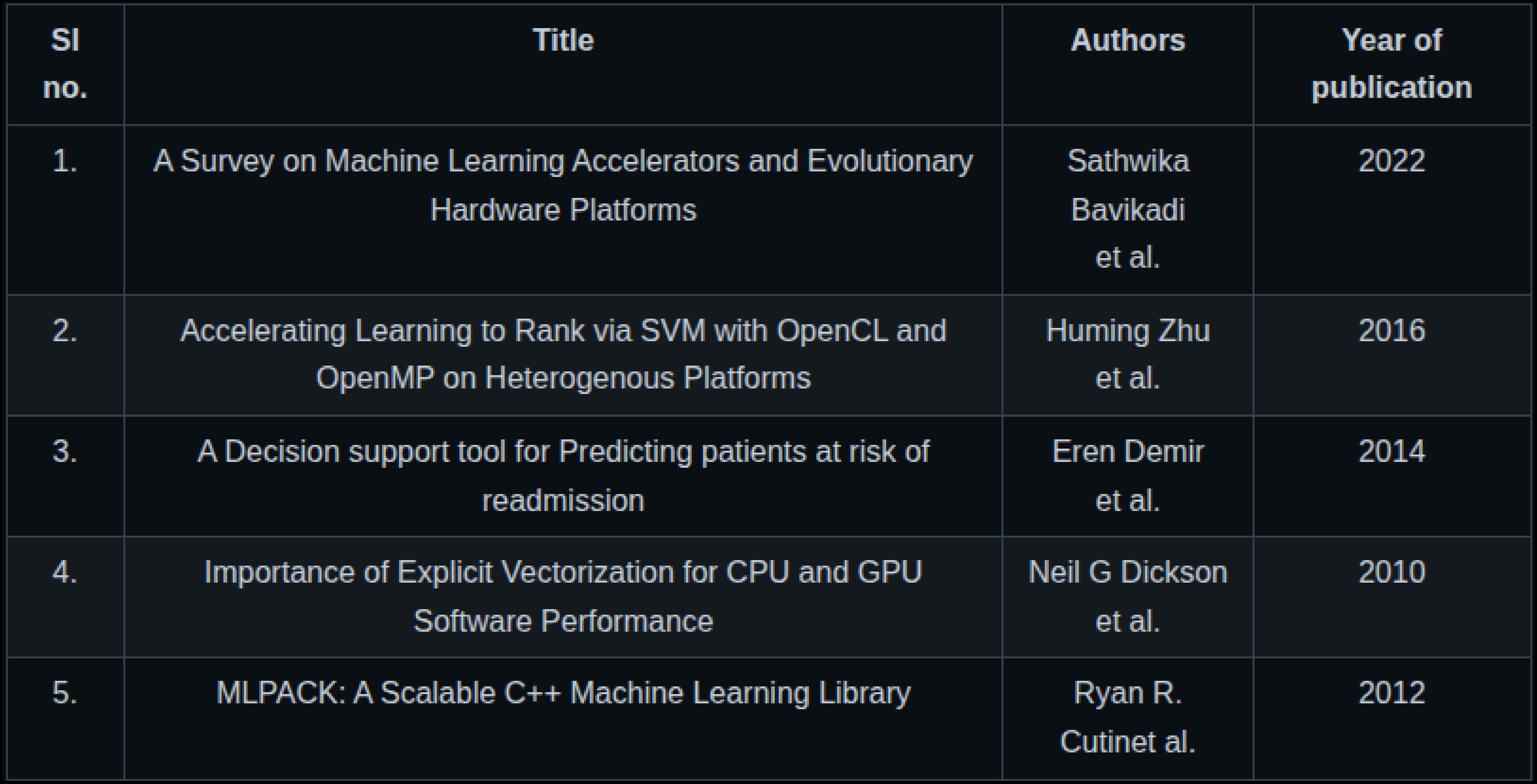
01. INTRODUCTION

Glacier.ML is a header only Supervised Machine Learning library, built entirely using C++.

Aims to be a lightweight, fast alternative to **Scikit-learn**

Currently houses five models, with 6 more in the roadmap. Classification models already benchmarked against Scikit-learn models, achieving comparable performance.

02. LITERATURE SURVEY



Sl no.	Title	Authors	Year of publication
1.	A Survey on Machine Learning Accelerators and Evolutionary Hardware Platforms	Sathwika Bavikadi et al.	2022
2.	Accelerating Learning to Rank via SVM with OpenCL and OpenMP on Heterogenous Platforms	Huming Zhu et al.	2016
3.	A Decision support tool for Predicting patients at risk of readmission	Eren Demir et al.	2014
4.	Importance of Explicit Vectorization for CPU and GPU Software Performance	Neil G Dickson et al.	2010
5.	MLPACK: A Scalable C++ Machine Learning Library	Ryan R. Cutinet al.	2012

03. EXISTING SYSTEM, FORMULATION OF PROPOSED SYSTEM

EXISTING SOLUTION - SCIKIT-LEARN

A Production-Grade Pythonic API:

- **High level**, **industry grade** Python API, used extensively throughout the **world.**
- Contains robust and easy documentation, enabling fast prototyping and useage.

Highly-Optimized C, Cython Core:

- Majorly uses NumPy and SciPy, which themselves are powered by C/C++, followed by Cython and C in performance critical regions and Joblib to implement parallelization.
- Code is highly optimized by numerous researchers and professionals over the years.

PROPOSED SYSTEM - GLACIER.ML

Self motivated hands-on initiative:

- Solo, systems-level high performance project to build core ML algorithms from scratch in C++.
- Aimed to gain deep, first-principles understanding of algorithmic design, memory management, and parallel computing through implementation.

Solo developer team:

- Allows me to play multiple roles required in an end-to-end software project, giving me invaluable hands-on experience.
- Allows me to use important professional tools and practices, which a conventional project would not provide.

Insights through implementation:

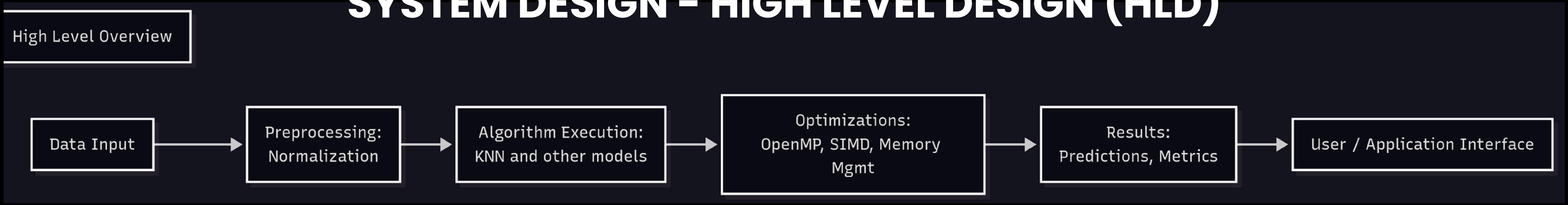
- Uncovers the "how" and "why" behind implementation and resulting performance.
- Initial naive implementations serve as a baseline, revealing performance bottlenecks and room for optimizations.

From naivete to Performance:

- Naive KNN began 200x slower than Scikit-learn, proving this was a ground-up, legitimate implementation
- OpenMP + SIMD optimizations pushed performance to near-competitive speed

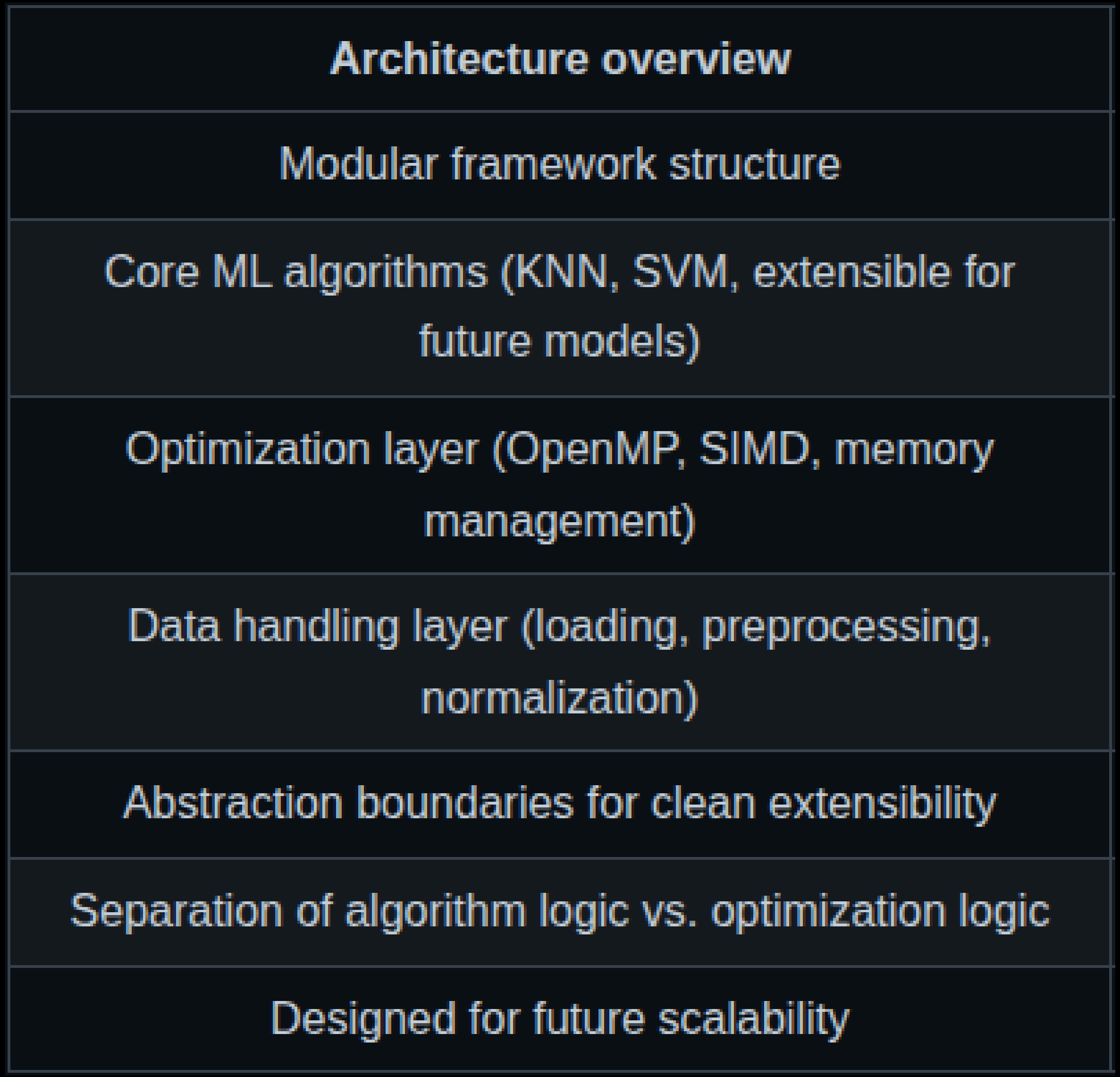
04. SYSTEM DESIGN

SYSTEM DESIGN - HIGH LEVEL DESIGN (HLD)

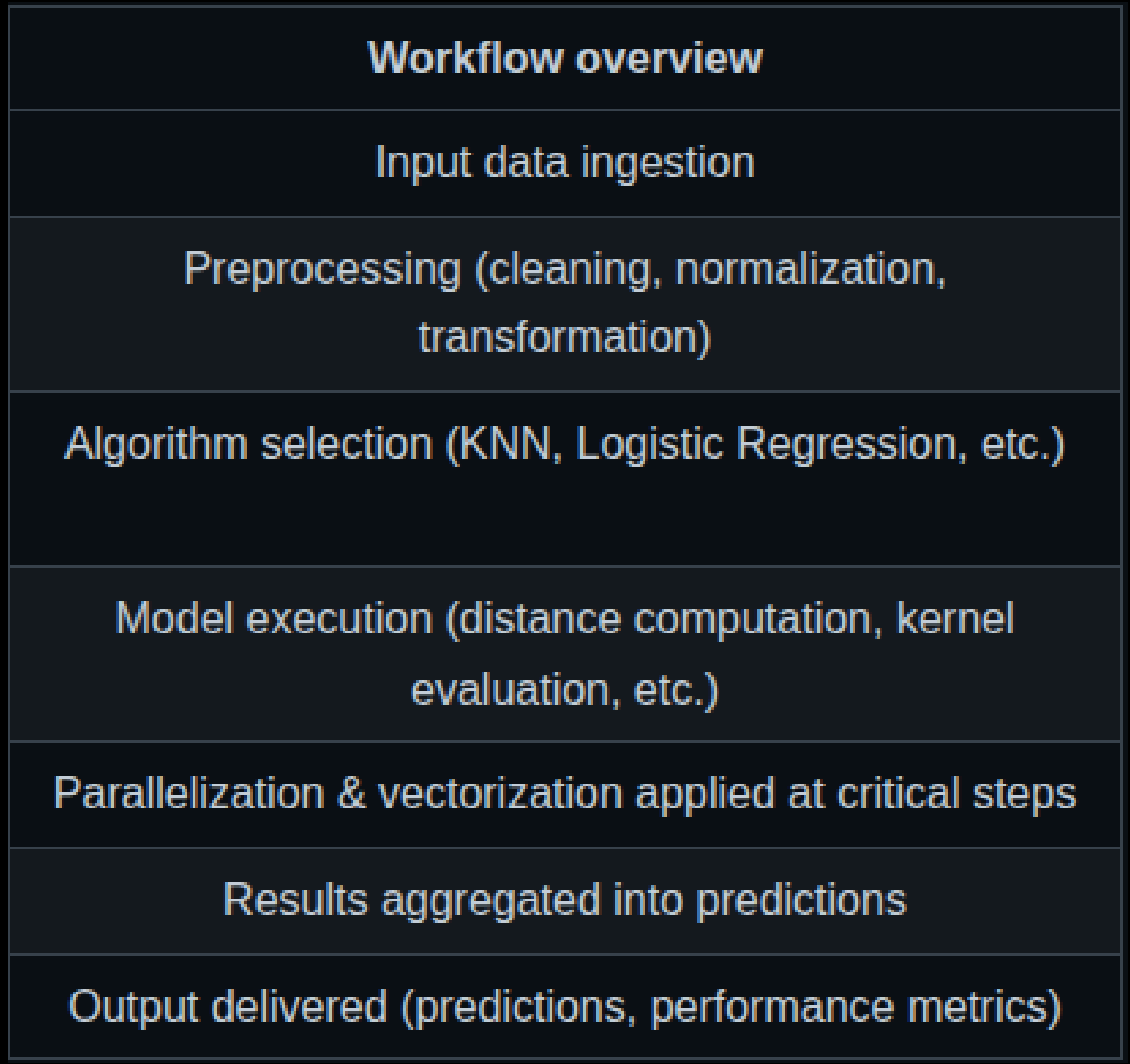


High Level Overview

Data Input → Preprocessing: Normalization → Algorithm Execution: KNN and other models → Optimizations: OpenMP, SIMD, Memory Mgmt → Results: Predictions, Metrics → User / Application Interface

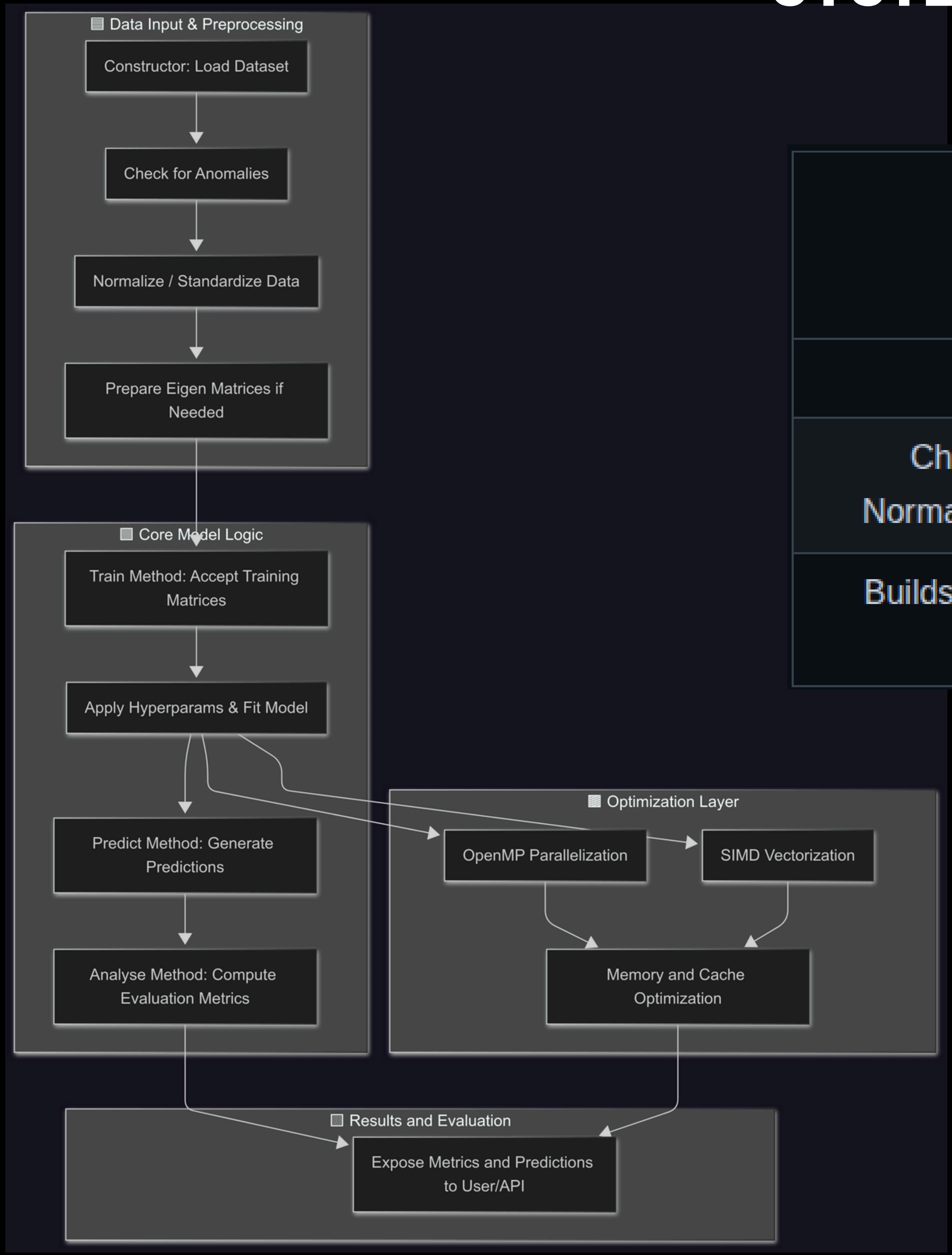


Architecture overview
Modular framework structure
Core ML algorithms (KNN, SVM, extensible for future models)
Optimization layer (OpenMP, SIMD, memory management)
Data handling layer (loading, preprocessing, normalization)
Abstraction boundaries for clean extensibility
Separation of algorithm logic vs. optimization logic
Designed for future scalability



Workflow overview
Input data ingestion
Preprocessing (cleaning, normalization, transformation)
Algorithm selection (KNN, Logistic Regression, etc.)
Model execution (distance computation, kernel evaluation, etc.)
Parallelization & vectorization applied at critical steps
Results aggregated into predictions
Output delivered (predictions, performance metrics)

SYSTEM DESIGN - LOW LEVEL DESIGN (LLD)


Data Input & Preprocessing
Constructor: Load Dataset
Check for Anomalies
Normalize / Standardize Data
Prepare Eigen Matrices if Needed
Core Model Logic
Train Method: Accept Training Matrices
Apply Hyperparams & Fit Model
Predict Method: Generate Predictions
Analyse Method: Compute Evaluation Metrics
Optimization Layer
OpenMP Parallelization
SIMD Vectorization
Memory and Cache Optimization
Results and Evaluation
Expose Metrics and Predictions to User/API

Data input & Pre-processing `constructor()`
Constructor loads data
Checks for anomalies in the dataset, Normalizes the data for easier computation
Builds the Eigen matrices for mathematical computation

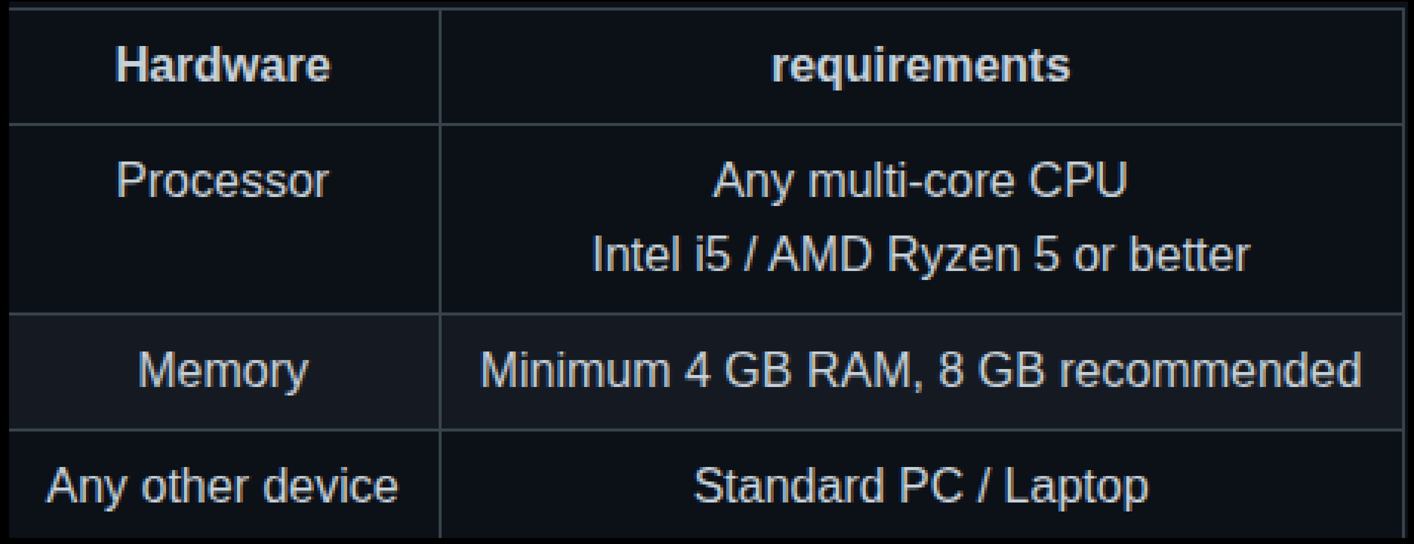
Core Model Logic `.train()`
Accepts hyper-parameters
Conducts model specific mathematical computations
Stores parameters and hyprer-parameters for predicting the results

Optimization layer `constructor()` & `.train()`
Uses cache-friendly flat matrices for memory optimization
Uses OpenMP to implement multithreading and SIMD instructions (vectorization)

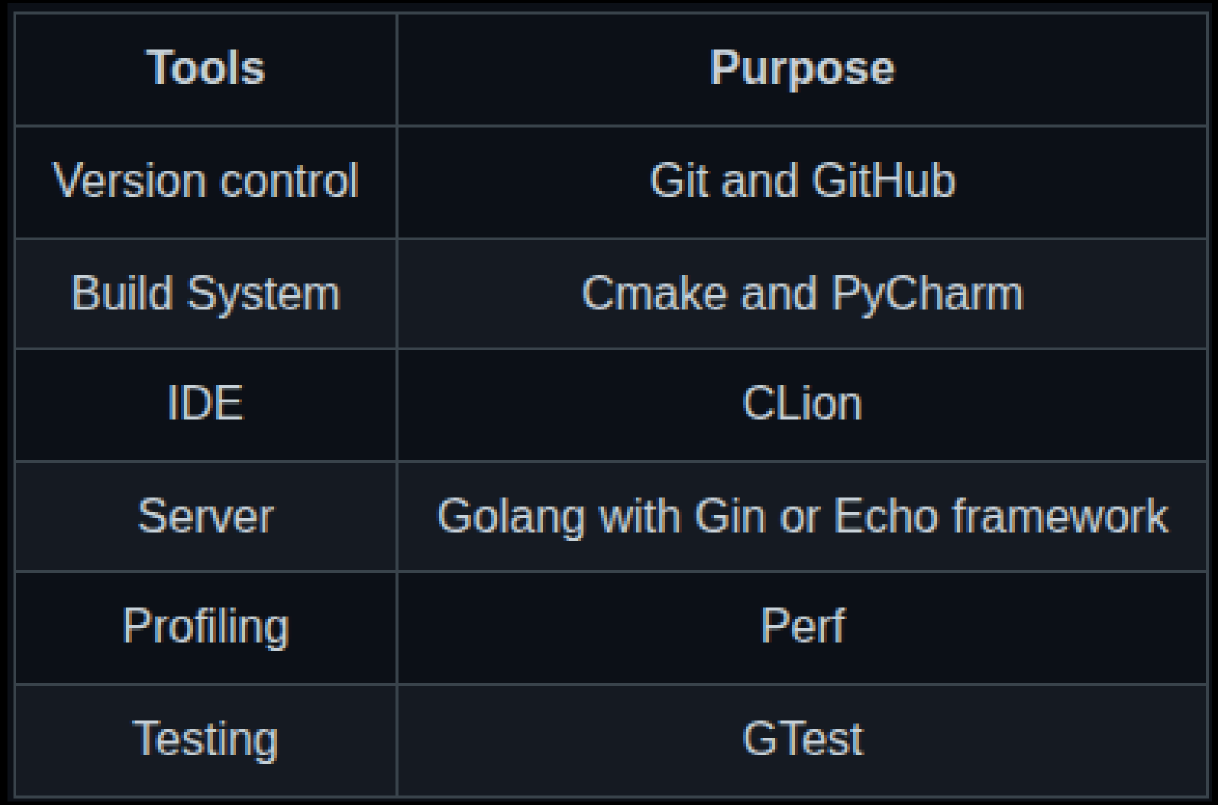
Results and Evaluation `.predict()` & `.analyze()`
Produces the result using stored parameters and hyper-parameters
Generates task specific evaluation metrics

05. SYSTEM REQUIREMENTS

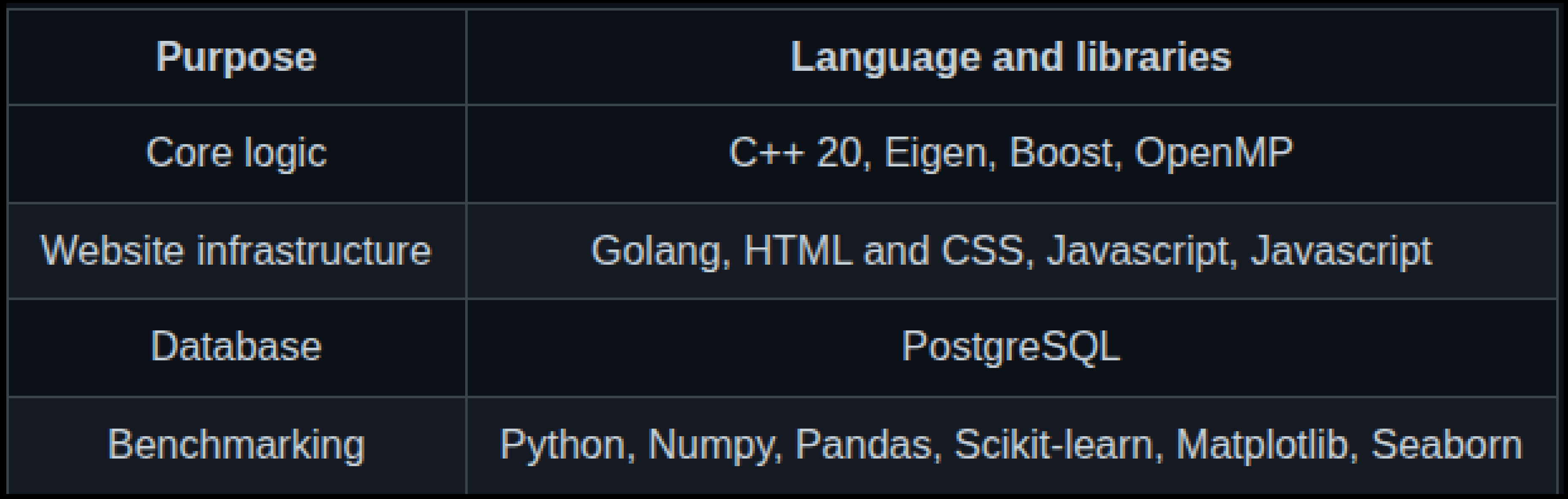
HARDWARE & SOFTWARE REQUIREMENTS



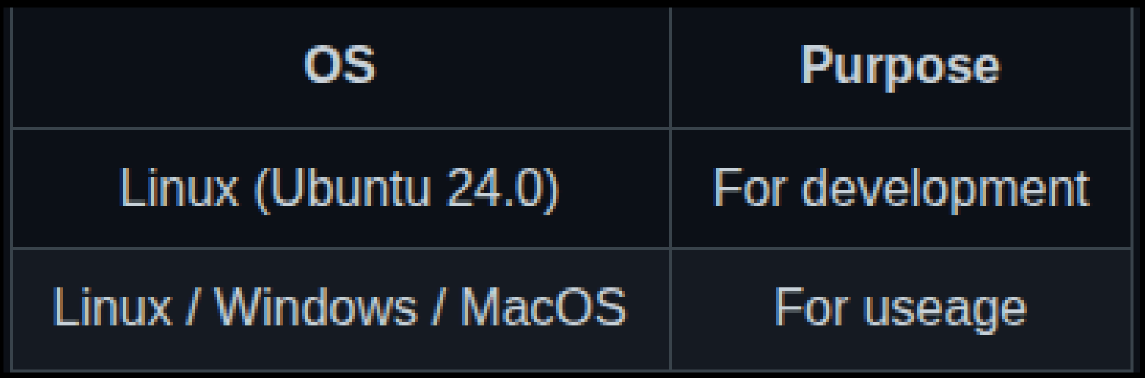
Hardware	requirements
Processor	Any multi-core CPU Intel i5 / AMD Ryzen 5 or better
Memory	Minimum 4 GB RAM, 8 GB recommended
Any other device	Standard PC / Laptop



Tools	Purpose
Version control	Git and GitHub
Build System	Cmake and PyCharm
IDE	CLion
Server	Golang with Gin or Echo framework
Profiling	Perf
Testing	GTest



Purpose	Language and libraries
Core logic	C++ 20, Eigen, Boost, OpenMP
Website infrastructure	Golang, HTML and CSS, Javascript, Javascript
Database	PostgreSQL
Benchmarking	Python, Numpy, Pandas, Scikit-learn, Matplotlib, Seaborn



OS	Purpose
Linux (Ubuntu 24.0)	For development
Linux / Windows / MacOS	For useage

06. CONCLUSION: FROM MATH TO SYSTEMS

- **Glacier.ML:** A solo, systems-level self driven project demonstrating a first-principles approach to machine learning infrastructure.
- **Validated Expertise:** My work on the KNN models prove that a deep understanding of memory management, parallelization, and low-level optimizations can yield significant performance gains.
- **Future-Ready:** This experience has equipped me with a unique, versatile skill set in algorithmic optimization and high-performance computing, which is essential for solving complex engineering challenges.

07. REFERENCES

- **Glacier_ML / KNNClassifier:** To understand the general model structure and to help design system design diagrams.
- **“Complete System Design Roadmap 2025” by Apna College:** To understand the concept of system design and it’s importance.
- **Google Scholar, ResearchGate, arXiv, IEEE Xplore:** To gather relevant research papers.

THANK YOU!