Low Latency Web API for Genetic Data Analysis

- 112103015 Atharva Mutsaddi T3 Div1

January 16, 2024

1 Introduction

Purpose: As the global internet penetration rate rises and the domains of artificial intelligence (AI) and big data analytics experience substantial expansion, the magnitude of data that websites must contend with, along with the surge in web traffic, has intensified. The inherent latency associated with APIs scripted in JavaScript, Python, and PHP has become a limiting factor, adversely impacting website performance. Hence, there is a need to formulate the architecture of low-latency APIs, leveraging lower-level programming languages such as C, C++, or Rust to provide enhanced computational efficiency, aligning with the evolving demands of a data-intensive and interconnected digital landscape. The design and performance of this API will be showcased through a Genetic Data Analyzer, harnessing the algorithmic prowess and efficiency inherent in C++.

2 Product Description

2.1 Perspective

Product aims to design a REST API in C++ using pre-existing libraries HTTP libraries which are compatible with all C++ versions greater than C++ 11. Main features to be included in this API will be centered around harnessing the speed and ease of designing algorithms in C++ with the help of custom and inbuilt optimized Data Structures and the use of a multi-threaded environment (which is lacking in JavaScript and Python). Analytics features will be provided in the form of Genetic Mutation Detection and K-Mer Analysis.

The designed API will be able to communicate with API calls in most languages like Python and JavaScript, making it integrable with most modern Front-End and middlewares.

2.2 Product Functioning

The analytical algorithms integral to the product will be meticulously crafted and implemented using the efficiency and robustness of C++. The product operates seamlessly by ingesting a text file, preferably in a plain text format, encompassing extensive genetic sequence data. This file is transmitted from the frontend, primarily developed in JavaScript, through a file upload mechanism. Subsequently, the C++ server processes the input, executing analytics algorithms, and delivers the corresponding analytical results in the form of a JSON file.

The generated JSON file serves as a structured output format, encapsulating comprehensive analytics-related data. This data is utilized to dynamically populate the Document Object Model (DOM) on the frontend. The use of JSON facilitates a standardized and easily interpretable format, and abides by REST principles.

The architecture is optimized for parallel processing, leveraging the multi-threaded capabilities of C++ to enhance computational speed and overall system performance. This not only ensures a responsive and seamless user experience but also positions the product as a robust and high-performance solution for genetic data analysis.

2.3 User Interaction

2.3.1 User Class:

Researchers and Students in Genetics: Individuals engaged in genetic research, including students pursuing academic studies in genetics.

2.3.2 Experience and Expertise:

Experience Level: Varied, ranging from novice researchers with basic knowledge to experienced professionals in the field of genetics. **Expertise:** Familiarity with basic genetic concepts and a general understanding of KMer Analysis and Genetic Mutation detection.

2.3.3 Technical Proficiency:

Proficiency Level: Basic to intermediate proficiency in using computer applications. Some familiarity with common data analysis tools.

2.3.4 Domain Knowledge:

Genetic Data Analysis Knowledge: Basic understanding of genetic data and common genetic analysis techniques. Familiarity with terms related to KMer Analysis and Genetic Mutation detection.

2.3.5 Usage Scenarios:

Typical Scenarios: Conducting preliminary analysis on genetic sequences for educational purposes. Exploring basic features of KMer Analysis and Genetic Mutation detection in a research setting.

2.3.6 Workflow:

Workflow: Uploading genetic data files. Initiating KMer Analysis and Genetic Mutation detection processes. Reviewing and interpreting analysis results.

2.4 Design Constraints:

1. The application must be compatible with Windows 10+, macOS 10.14+, and Ubuntu 18.04+.
2. The product must be delivered and ready for deployment according to academic calendar.
3. Single member in the team

2.5 Assumptions and Dependencies

2.5.1 Assumptions:

1. There is agreement among stakeholders regarding the project scope, objectives, and requirements.
2. Project complies with relevant laws, regulations, and industry standards unless explicitly stated otherwise.
3. Input data provided for testing and development is accurate, representative, and of sufficient quality for illustrating performance enhancement via low latency API design in C++

2.5.2 Dependencies:

1. External Open Source HTTP Web server libraries for C++
2. C++ version greater than 11
3. Genetic Data from open source data sets from NCBI
4. Project Timeline Dependencies
5. Testing Environments in JavaScript or Python for performance and accuracy testing.

2.6 External Interface Requirements:

1. Modern React UI/UX libraries/ framework for web based software interface
2. HTTP and REST API communication standards via JavaScript libraries