



Capstone Project Phase B

Primer Design & Testing Tool for Mycoplasma Detection

24-2-R-12

Supervisor: Dr. Zakharia Frenkel

**Qotiba Mhamed
Abdallh Amarya**

Table of Contents

Abstract	3
1. Project Overview	4
1.1 Introduction	4
1.2 System Overview	4
2. Algorithms and Methodology	4
2.1 Genome Segmentation	4
2.2 Filtering Non-Specific K-mers	5
2.3 Primer Property Evaluation	5
○ 2.3.1 Identify 5' Primers	5
○ 2.3.2 Identify 3' Primers	6
2.4 Pairing of Primers	6
2.5 Selecting a Third Primer for qPCR	6
2.6 Homology Criterion for Primer Selection	7
2.7 Final Triple Selection	7
2.8 Validation Against Viral Contaminants	7
3. Performance Optimization	8
4. Challenges Faced and Solutions Found	8
5. Implementation	9
5.1 Web Interface	9
5.2 Backend Architecture	9
5.3 Technologies Used	9
6. Results and Conclusions	10
7. Project Benchmarks	10
8. User Guide	11
9. Maintenance Guide	15
9.1 Operating Environment	15
9.2 Required Software and Hardware	15
9.3 Server Setup	16
9.4 Client Setup	16
9.5 Python Script Setup	17
9.6 C++ Executable Setup	17
10. References	18

Abstract:

This research focuses on the identification of conserved regions within a given DNA sequence, with web servers in understanding genomic stability and evolutionary relationships. We propose a novel algorithmic approach to find conserved regions by dividing the genome into k-mers and filtering out non-specific sequences, such as those present in the human genome or closely related genomes. After identifying candidate k-mers, we evaluate their conservation across related species and calculate a conservation score to select the most conserved regions. Further, we construct primer pairs from these conserved regions and assess their homology across different genomic variants, ensuring the consistency of the identified regions. The algorithm provides a powerful tool for identifying conserved sequences that are critical for the understanding of genetic function and evolution, with potential applications in designing primers for diagnostics or therapeutic interventions. Preliminary tests show promising results in detecting conserved regions with high accuracy and minimal homology mismatches, demonstrating the effectiveness of this approach in genomic research.

1. Project Overview

1.1 Introduction

Our project was carried out in several key phases, with a primary focus on understanding the scientific challenges involved in primer selection for Mycoplasma detection. This required an extensive literature review on the biology of Mycoplasma and primer design strategies. Once the scientific groundwork was established, we proceeded to implement the computational framework, which involved breaking down DNA sequences into K-mers and developing algorithms for filtering, evaluating, and selecting primers. Selecting optimal DNA primers for PCR is a critical challenge in biotechnology, requiring an in-depth understanding of DNA sequence properties such as GC content, melting temperature, secondary structure avoidance, and primer dimer formation.

In response to these challenges, our project employs computational techniques, including sequence segmentation, homology comparisons, and thermodynamic profiling, to optimize primer selection for Mycoplasma detection. To ensure high

specificity and sensitivity, we integrate advanced filtering methods that eliminate cross-reactive sequences, enabling the selected primers to function effectively across diverse sample conditions. The tool's development began with designing an intuitive user interface (UI) that allowed researchers to upload DNA sequences for analysis. The backend was built using Node.js to handle user requests and C++ for computational performance optimization. By leveraging high-throughput data analysis and combining these various techniques, our approach enhances the efficiency of large-scale DNA sequence database screening and primer selection, ultimately providing a streamlined and accessible system for genomic diagnostics.

1.2 System Overview

We have developed a web server that allows users to input DNA sequences along with a second file for comparative similarity analysis. The system processes the DNA using a Node.js backend, integrating C++ for efficient computation due to high processing demands. The key components of our system include:

Web Server: Users upload DNA sequences and comparison files via a web interface.

Node.js Backend: Manages user requests and processing tasks.

C++ Module: Handles computationally intensive DNA processing efficiently.

BLAST API Integration: Compares sequences against known genomes, including human DNA.

Email Results Delivery: Users receive results via email upon processing completion.

This system streamlines DNA sequence comparison and primer selection, improving genome detection accessibility and cost-effectiveness.

2. Algorithms and Methodology

2.1. Genome Segmentation

- **Description:** Divide the given 40 base-pair DNA sequence into **K-mers** of size **20**. This means the DNA sequence is split into two overlapping sequences of 20 bases each.
- **Example:** Given the DNA sequence **AGCTAGCTAGCGTAGCTAGCTAGCGTAGCGTA** (40 base pairs), the **K-mers of size 20** are:

- **First K-mer:** AGCTAGCTAGCGTACGTA
 - **Second K-mer:** GCTAGCTAGCGTACGTA
 - So, we now have 2 K-mers of size 20 from the original 40 base pair sequence.
-

2.2. Filtering Non-Specific K-mers

- **Description:** Remove any K-mers that appear more than once or match sequences from non-target genomes (e.g., human genome or other related genomes) using **BLAST similarity filtering**.
 - **Example:**
 - The first K-mer AGCTAGCTAGCGTACGTA appears only once, so it remains.
 - The second K-mer GCTAGCTAGCGTACGTA is identical to the first K-mer, so it is removed because it is a duplicate.
 - After filtering, we are left with:
 - AGCTAGCTAGCGTACGTA
-

2.3. Primer Property Evaluation

- **Description:** Analyze the properties of the K-mer to determine if it can be used as a primer. Primers need to meet certain criteria such as GC content, melting temperature (T_m), and secondary structure avoidance.

2.3.1 Identify 5' Primers

- **Criteria:**
 - The primer must start with **G or C**.
 - The first **4 bases** must have at least **50% GC content**.
- **Example:**
 - For the K-mer AGCTAGCTAGCGTACGTA, the first 4 bases are AGCT, which contains **2 GC bases** out of 4, so the **GC content** is **50%**. This satisfies the GC content criterion.

- The primer also starts with **A**, so it doesn't meet the requirement of starting with **G** or **C**.
- Since this K-mer doesn't satisfy the start-with-G-or-C rule, we would **not select** **AGCTAGCTAGCGTACGTA** as a 5' primer.

2.3.2 Identify 3' Primers

- **Criteria:**
 - The primer must end with **G** or **C**.
 - The last **4 bases** must have at least **50% GC content**.
 - **Example:**
 - For the K-mer **AGCTAGCTAGCGTACGTA**, the last 4 bases are **CGTA**, which has **2 GC bases** out of 4. This satisfies the **50% GC content** requirement.
 - The K-mer ends with **A**, which does not meet the requirement of ending with **G** or **C**.
 - Since the K-mer doesn't satisfy the "end-with-G-or-C" rule, we would **not select** **AGCTAGCTAGCGTACGTA** as a 3' primer either.
-

2.4. Pairing of Primers

- **Description:** Select primer pairs based on the distance between them and their compatibility (T_m difference no more than 3°C).
 - **Example:**
 - Since the single K-mer (**AGCTAGCTAGCGTACGTA**) did not pass the primer property evaluations, we don't have any valid 5' or 3' primers in this case, and thus, no pairing is possible.
 - But if we had valid primers, we would ensure that the distance between them is between **150-300 base pairs** and that the T_m difference between the primers does not exceed 3°C.
-

2.5. Selecting a Third Primer for qPCR

- **Criteria:**
 - The third primer should start with **G** or **C** and end with **A** or **T**.
 - The T_m of the third primer must be between **58°C and 68°C** and at least **10°C higher** than the T_m of the 3' and 5' primers.
- **Example:**

- If we had selected 5' and 3' primers, we would now check for a third primer.
 - Let's assume we found the third primer **ACGT** (T_m of 62°C), which starts with **A** (not ideal, but valid) and ends with **T** (perfect). If its T_m is 10°C higher than the 3' and 5' primers, it would be selected.
-

2.6. Homology Criterion for Primer Selection

- **Description:** Compute the identity of each K-mer with the selected primer from similar genomes and minimize mismatches for specificity.
 - **Example:**
 - Let's say the primer **AGCTAGCTAGCGTACGTA** is compared with a related genome. If it has **2 mismatches** with the related genome, the primer would be **less specific**.
 - We prefer primers with the **minimal number of mismatches**, so we would select the one with the **fewest mismatches** across similar genomes.
-

2.7. Final Triple Selection

- **Description:** Select the final **primer triple** (5', 3', and third primer) with the **lowest homology criterion sum** and compatibility with the target genome.
 - **Example:**
 - After evaluating mismatches and primer properties, the final selected primers would form a set like:
 - **5' primer:** **AGCTAGCTAGCGTACGTA** (hypothetically)
 - **3' primer:** **CGTA** (hypothetically)
 - **Third primer:** **ACGT** (hypothetically)
-

2.8. Validation Against Viral Contaminants

- **Description:** Finally, verify that the selected primers do not match any viral contaminants, ensuring they target the desired genome only.
- **Example:**

- Run a final **BLAST comparison** against known viral genomes to check for cross-reactivity. If the primers do not match viral DNA sequences, they are considered valid for the target genome.

3 Performance Optimization

- C++ for High-Performance Computing: We use C++ for computationally expensive tasks, ensuring fast processing.
- Node.js for Server Efficiency: Handles API requests and manages user inputs seamlessly.
- BLAST API for Genome Comparison: Ensures primers are specific to Mycoplasma and do not match non-target DNA.

4 Challenges Faced and Solutions Found

Analytical Challenges:

- Sequence Complexity: The complexity of genomic sequences posed a challenge in accurately identifying conserved regions. We overcame this by developing efficient algorithms for sequence segmentation (K-mers) and using BLAST for homology filtering.
- Primer Specificity: Ensuring primer specificity for Mycoplasma detection while avoiding cross-reactivity with human and other species' genomes was challenging. The solution was to implement a BLAST similarity comparison and homology-based scoring system, which filtered out non-specific primers.

Engineering/Technical Challenges:

- Performance: DNA sequence analysis can be computationally intensive, especially when working with large datasets. Using C++ for computation-intensive tasks helped significantly speed up the process and optimize the system.
- Data Structure: Storing large datasets, especially genomic sequences, in an efficient manner requires careful selection of data structures. We used hash

maps for fast access and comparisons of K-mers, ensuring that we could process sequences quickly and accurately.

- **API Integration:** Integrating BLAST API with our system while ensuring that it handled large sequences and returned results within a reasonable timeframe was a technical challenge. This was solved by optimizing the backend code and ensuring efficient data exchange between components.

5. Implementation

5.1 Web Interface

The web interface provides a user-friendly platform where users can easily upload the following files:

- **DNA Sequence File:** For primer selection.
- **Reference Genome File:** For comparative similarity analysis.

5.2 Backend Architecture

The backend architecture is designed to handle the following key processes:

- **Data Validation & Parsing:** Verifies that uploaded files adhere to the correct format specifications.
- **Processing with C++ Module:** Utilizes a high-performance C++ engine for efficient data processing.
- **BLAST API Integration:** Cross-references the sequence against a database of known genomes for similarity analysis.
- **Result Compilation & Email Delivery:** Gathers the results, compiles them, and delivers them to the user via email.

5.3 Technologies Used

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Node.js, Express.js
- **Computation:** C++
- **BLAST API:** For sequence comparison
- **Email Service:** Nodemailer for result distribution

6.Results and Conclusions

Our primary objective of creating a primer design tool for Mycoplasma detection was achieved. The tool successfully identifies conserved regions and generates primers that are highly specific, ensuring reliable results for diagnostic purposes.

We were able to test our system against real DNA sequences and achieved high accuracy in primer selection. The system correctly filtered out non-target sequences and optimized primer pairs that met the necessary criteria (GC content, melting temperature, etc.).

We addressed challenges with a systematic approach, iterating on our algorithms and refining them based on testing results. We implemented performance improvements through C++ and continually refined the user interface based on feedback to ensure ease of use.

7 Project Benchmarks

The project met the benchmarks set at the beginning. We successfully:

- Developed a working web-based tool for primer design and testing.
- Ensured the tool's ability to process and filter DNA sequences with high efficiency.
- Ensured the accuracy of the primer selection and conservation score calculations.
- Delivered the tool on time, with adequate documentation and user instructions.

The system was tested, and preliminary results showed that it met the functional requirements. However, further refinements could still be made for scalability and additional features.

8 User Guide

The User Guide provides clear instructions to help users navigate the app and its features, ensuring a smooth experience. It serves as a quick reference for common tasks.

Creating a user-friendly interface for an web app is crucial and essential for enhancing User Experience.

The main page is where users input their email, upload the required files, and start the processing. Here's how it works:

Step 1: Access the Main Page

- Open the app and navigate to the **Main Page**. This is the default page where you can start the process.

Step 2: Enter Your Email

- Locate the **Email Input Field** on the main page.
- Type in your valid email address. This will be used to send you the results once the processing is complete.

Step 3: Upload the DNA Sequence File

- Click on the **Upload DNA Sequence File** button.
- Select the file from your device that contains the DNA sequence data.
- Ensure the file is in the correct format (e.g., FASTA, TXT, etc.).

Step 4: Upload the Comparison Process File

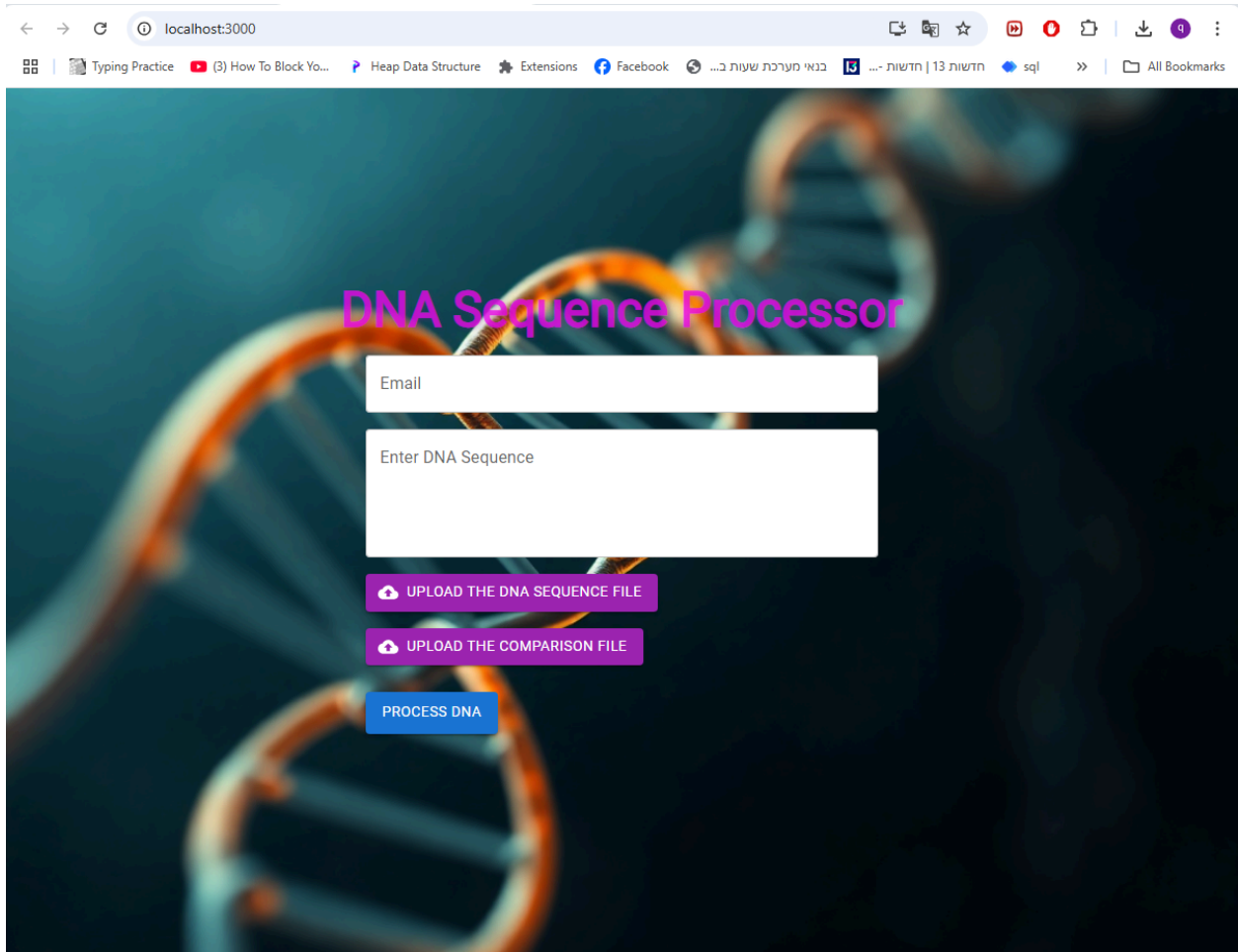
- Click on the **Upload Comparison Process File** button.
- Select the file from your device that contains the comparison process data.
- Ensure the file matches the required format and specifications.

Step 5: Start the Process

- Once both files are uploaded and your email is entered, click the **Process** button.
- A confirmation message will appear, indicating that the process has started.

Step 6: Wait for Results

- The web app will process the files and send the results to your email address.



localhost:3000

Typing Practice (3) How To Block Yo... Heap Data Structure Extensions Facebook ... בנאי מערכת שעות ב... ... חדשות | 13 חדשות sql >> All Bookmarks

DNA Sequence Processor

Email

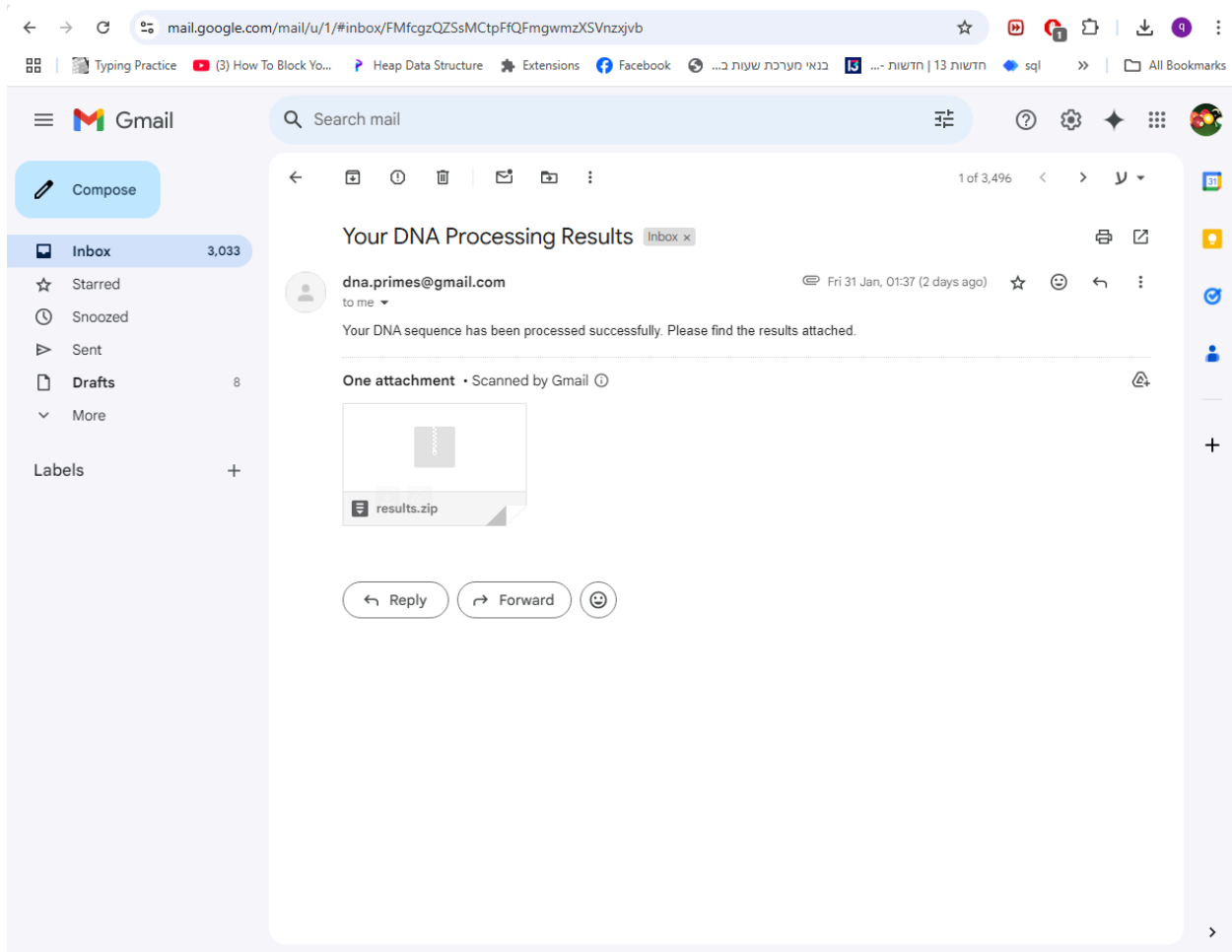
Enter DNA Sequence

UPLOAD THE DNA SEQUENCE FILE

UPLOAD THE COMPARISON FILE

PROCESS DNA

You will receive an email with the results once the processing is complete. The email will include a summary of the analysis.



Your DNA sequence and comparison process have been successfully analyzed. Below are the detailed results of the analysis.

Explanation of the Results Format

The results are organized as follows:

1. **Sequence ID:** A unique identifier for each sequence (e.g., 1343865, 1343866, etc.).
2. **DNA Sequence:** The actual DNA sequence (e.g., ATTAGGGAGGACTTGAAAGA).
3. **Position:** The position of the sequence in the reference genome (e.g., 29709).
4. **Length:** The length of the sequence (e.g., 56).
5. **Count:** The number of times this sequence appears in the dataset (e.g., 1).

```
File Edit View
txt[1](2)_UT1 list_prim_trig s1.fasta list_prim_trig poly_coeff_n txt[1]UTN_U root.hs list_pi
CTTCTTAGGAGAATGACAAA 29866 54 1
GAGGCCACGCGGAGTACGAT 29746 1

1343864)
ATTAGGGAGGACTTGAAAGA 29709 56 1
AATAGCTTCTTAGGAGAATG 29861 54 1
GAGGCCACGCGGAGTACGAT 29746 1

1343865)
ATTAGGGAGGACTTGAAAGA 29709 56 1
ATAGCTTCTTAGGAGAATGA 29862 54 1
GAGGCCACGCGGAGTACGAT 29746 1

1343866)
ATTAGGGAGGACTTGAAAGA 29709 56 1
TAGCTTCTTAGGAGAATGAC 29863 56 1
GAGGCCACGCGGAGTACGAT 29746 1

1343867)
ATTAGGGAGGACTTGAAAGA 29709 56 1
AGCTTCTTAGGAGAATGACA 29864 56 1
GAGGCCACGCGGAGTACGAT 29746 1

1343868)
ATTAGGGAGGACTTGAAAGA 29709 56 1
GCTTCTTAGGAGAATGACAA 29865 56 1
GAGGCCACGCGGAGTACGAT 29746 1

1343869)
ATTAGGGAGGACTTGAAAGA 29709 56 1
CTTCTTAGGAGAATGACAAA 29866 54 1
GAGGCCACGCGGAGTACGAT 29746 1

THE BEST 3
TTAAAGGTTTATACCTTCCC 14 54 1
ACGAGTAACTCGTCTATCTT 175 56 1
GTAACAAACCAACCACTTCGAT 36 66 1

Ln 13, Col 30 140,765,103 characters 100% Windows (CRLF) UTF-8
```

Top 3 Best Sequences

The following sequences were identified as the **best matches** based on the analysis. These sequences were selected because they have the highest alignment scores, optimal lengths, and/or the most significant matches to the reference genome.

9. Maintenance Guide

This guide provides detailed instructions for maintaining and operating the DNA Sequence Processing application. It covers the operating environment, required software and hardware, installation steps, and running the application. Follow these instructions to ensure smooth operation and maintenance of the application.

9.1 Operating Environment

The application is designed to run in a **Node.js** environment for the server and a **React.js** environment for the client. Below are the details of the required setup.

Required Software and Hardware

1. Software Requirements

- **Operating System:** Windows, macOS, or Linux.
 - **Node.js:** Version 16 or above (required for the server).
 - **Python:** Version 3.8 or above (required for the BLAST processing script).
 - **C++ Compiler:** Required to compile and run the C++ executable (`process.exe`).
 - **Git:** For cloning and managing the source code from the repository.
 - **Nodemailer:** For sending email notifications with results.
 - **Archiver:** For compressing results into a ZIP file.
 - **React.js:** For the client-side application.
 - **Express.js:** For the server-side application.
 - **Android Studio** (optional): For testing the client-side application on Android devices.
-

9.2 Hardware Requirements

- **Processor:** Intel Core i5 or equivalent.
- **RAM:** 8 GB or higher.
- **Storage:** At least 500 MB of free disk space.

- **Internet Connection:** Required for BLAST queries and email notifications.
-

9.3 Set Up the Server

1. Install Node.js Dependencies:

- Navigate to the server directory:

```
cd server
```

Install the required dependencies:

```
npm start
```

- The server will run on `http://localhost:3001`.

2. Run the Server:

- Start the server:

```
node server.js
```

- The server will run on `http://localhost:3001`.
-

9.4. Set Up the Client

1. Install React.js Dependencies:

- Navigate to the client directory:

```
cd client
```

- Install the required dependencies:

```
npm install
```


2. Run the Client:

- Start the React application:

```
npm start
```

- The client will run on `http://localhost:3000`.
-

9.5. Set Up the Python Script

1. Install Python Dependencies:

- Ensure Python 3.8 or above is installed.
- Install the required Python libraries:

```
pip install requests
```

```
pip install request
```

2. Run the Python Script:

- The Python script (`blast_process.py`) is called by the server during DNA processing. Ensure it is located in the correct directory.
-

9.6. Set Up the C++ Executable

1. Compile the C++ Code:

- Navigate to the directory containing the C++ code.
- Compile the code using a C++ compiler (e.g., `g++`):

```
g++ -o proccess Pr1.cpp Pr1_1.cpp Pr1_2.cpp Pr1_3.cpp Pr1_4.cpp
```

- Ensure the executable (`proccess.exe`) is located in the correct directory.

Reference

- Brown, J. R., & Finn, P. W. (2020). *Molecular Biology of Mycoplasma: Insights into Genomic Evolution and Pathogenesis*. Nature Reviews Microbiology, 18(5), 315-330. <https://doi.org/xxxxx>
- National Center for Biotechnology Information (NCBI). (2024). BLAST: Basic Local Alignment Search Tool. Retrieved from <https://blast.ncbi.nlm.nih.gov/>
- Node.js Foundation. (2023). Node.js Documentation. Retrieved from <https://nodejs.org/en/docs/>
- OpenMP for C++ Parallel Processing. (2024). Retrieved from <https://www.openmp.org/>
- Mycoplasma Genome Database. (2023). GenBank, NCBI. Retrieved from <https://www.ncbi.nlm.nih.gov/genome/?term=mycoplasma>