2021-08

Sanger sequence automatic analysis tool development

Mero, Victor

NM-AIST

https://doi.org/10.58694/20.500.12479/1580

SANGER SEQUENCE AUTOMATIC ANALYSIS TOOL DEVELOPMENT

Victor Mero

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Master's in Information and Communication Science and Engineering of the Nelson Mandela African Institution of Science and Technology

Arusha, Tanzania

August, 2021

ABSTRACT

The Sanger sequencing technique still remains the cornerstone methods for the Deoxyribonucleic acid (DNA) sequencing. This is due to its high accuracy in targeting smaller genomic regions in a large number of samples, sequencing of variable regions and validating results of other DNA sequencing platforms such as those from next-generation sequencing studies. The analysis of Sanger sequence DNA data is computationally intensive requiring efficient and high computational power software tools. The most preferred tools are the proprietary licensed tools since they offer user-friendly interface and they offer more DNA analysis functionalities. However, the affordability of the tools may be limited especially for individual researchers or students due to their expensive licenses. Nevertheless, free and open-source licensed tools are available and but are not user-friendly, have steep learning curve since some lack graphical user interface, operating system platform dependent and have limited functionalities. This study presents a Sanger Sequence Automatic Analysis Tool (SSAAT), a software tool designed and implemented as a web application that provides a user-friendly graphical interface such as those provided in proprietary tools. The tool has abilities to extract raw data from sequence AB1 files, make base-calls, plot chromatogram, polymorphism detection, sequence alignment and report generation. Furthermore, the tool is free and open-source that can be easily accessed online through standard web browser applications. With the above-mentioned features, SSAAT can be used by molecular biologist as an alternative to proprietary tools and get comparable experience and DNA sequence analysis results.

DECLARATION

I, Victor Mero do hereby declare to the senate of the Nelson Mandela African Institution of Science and Technology that this dissertation is my own original work and that it has neither been submitted nor being currently submitted for degree award in any other institution.

Victor Mero	VMero	05/08/2021
Candidate Name	**Signature**	**Date**

The above declaration is confirmed

Dr. Dina Machuve	DMA	05/08/2021
Supervisor Name	**Signature**	**Date**

Dr. Jean-Baka Domelevo Entfellner	J.B.Entfellner	05/08/2021
Supervisor Name	**Signature**	**Date**

COPYRIGHT

CERTIFICATION

The undersigned certify that they have read the dissertation titled: *"Sanger sequence automatic analysis tool development"* and recommended for examination in fulfilment of the requirements for the degree of Master's in Life Sciences of the Nelson Mandela African Institution of Science and Technology.

Dr. Dina Machuve		05/08/2021
Supervisor Name	**Signature**	**Date**

Dr. Jean-Baka Domelevo Entfellner		05/08/2021
Supervisor Name	**Signature**	**Date**

ACKNOWLEDGMENT

First and foremost, I give thanks to the Almighty God for His protection, grace and throughout my studies. He has seen me through the end of my studies at the Nelson Mandela African Institution of Science and Technology (NM-AIST).

I highly appreciate the contribution of my supervisors, Dr. Dina Machuve, the internal supervisor and Dr. Jean-Baka Domelevo Entfellner from Biosciences eastern and central Africa (BecA-ILRI) Nairobi, Kenya. They both provided invaluable guidance throughout this research. I acknowledge the support for the short research attachment at the Biosciences eastern and central Africa (BecA-ILRI) hub in Nairobi campus for my research work.

I am extremely grateful to my parents and my young sisters, Sarah and Agnes for their love, caring support and encouragement throughout my studies period. I am forever grateful for the support from my lovely wife Shose and our son Shawn. They have given me love, prayers, sacrifices, patience, and support during all the time of my studies.

My friends and colleagues at NM-AIST and the management of NM-AIST especially those from CoCSE laboratory had a significant and important role during my studies and I thank you all.

DEDICATION

This work is humbly dedicated to my family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS AND SYMBOLS

A	Adenine
AB1	Applied Biosystems File Format
ASAP	Automated Sanger Analysis Pipeline
BecA-ILRI	Bioscience Eastern and Central Africa, International Livestock Research Institute
BLAST	Basic Local Alignment Search Tool
C	Cytosine
CLI	Command-Line Interface
CMD	Command Prompt
CoCSE	Computational and Communication Sciences and Engineering
CPAN	Comprehensive Perl Archive Network
CPU	Central Processing Unit
CRAN	Comprehensive R Archive Network
CSS	Cascading Style Sheet
DNA	Deoxyribonucleic Acid
EMBL	European Molecular Biology Laboratory
EMBOSS	European Molecular Biology Open Software Suite
FASTA	Fast-All File Format
G	Guanine
GUI	Graphical User Interface
HCI	Human–Computer Interaction
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
IP	Internet Protocol
NCBI	National Center for Biotechnology Information
Next-Gen	Next Generations Sequencing
NM-AIST	Nelson Mandela African Institution of Science and Technology
R	Programming Language for Statistical Computing and Graphics
SSAAT	Sanger Sequence Automatic Analysis Tool
T	Thymine
TCP	Transmission Control Protocol
UML	Unified Modeling Language

CHAPTER ONE

INTRODUCTION

1.1 Background of the Problem

Deoxyribonucleic Acid (DNA), is the genetic material found in almost all living organisms that include single-celled microorganisms, and multicellular mammals like human beings. The DNA are the information molecules of cells containing the genetic code, all messages that determine traits of organisms. It is found in the cell nucleus where it is named nuclear DNA, merely a small amount of DNA can as well be found in the cell's structures known as mitochondria that converts the energy from food into a form that cells can use.

The hereditary material in DNA is stored as a cipher made up of four nucleobases: adenine (A), guanine (G), cytosine (C), and thymine (T). The human DNA consists of about 3 billion bases, and only 1 % of those nucleobases are the different in all human population. The sequence of these nucleobases determines the building information for a living thing. The order is similar to how the alphabetical letters appear in a certain arrangement to compose words and sentences.

Deoxyribonucleic Acid bases pair up with each other, A with T and C with G, to form units called base pairs. Each base is also attached to a sugar molecule and a phosphate molecule. Together, a base, sugar, and phosphate are called a nucleotide. The assembly of the double helix is slightly comparable to a ladder, with the nucleobase pairs forming the ladder's step and the phosphate and sugar molecules making the vertical side parts of the ladder. Each strand of DNA in the double helix can serve as a pattern for reproducing the sequence of other nucleobases.

There are several areas of which DNA studies contribute to and among them are genetics and medical research. Since the discovery of DNA, the capability to diagnose diseases at an early stage has been extremely improved. Furthermore, DNA technologies have provided better ways to evaluate animals and human's genetic vulnerability to specific diseases. In this regard, pharmaceuticals have paved their way to develop new drugs to treat these diseases. Drugs can now be custom produced to match individual human biochemistry and genetic makeup. Some of diseases were previously considered deadly and their treatment remained in vain for a long time, however, through the DNA technologies innovations has speed up the advancement of drugs that could cure those diseases.

Though the invention of DNA has influenced the advancement of medicine mostly, its impact to other industries is still noteworthy. Fatherhood cases have a massive impact on families around the globe. Additionally, the use of DNA assessment has help to identify the paternity of children of which has an outstanding result to both their upbringing and their lives.

Deoxyribonucleic Acid has been notably important to the field of forensic science (Van-Oorschot *et al*., 2010). The DNA technology advancement has also improved the forensic investigation whereby the guilt or innocence of a victim being investigated for a wrongdoing can be determined (Roach, 2010). It likewise implies that uncommon evidence can still produce important signals regarding the committer of a crime. Similarly, the identification of victims can occur, particularly in scenarios where the victim's condition is unrecognizable to relatives (Taylor & Kieser, 2016). In this sense, DNA has been important in revolutionizing the entire field of forensic science (Taylor & Kieser, 2016; Van-Oorschot *et al*., 2010).

Determination of the order of DNA nucleotides in biological samples is known as DNA sequencing. This involves the use of scientific methods or technology to determine the order of the four bases: adenine (A), guanine (G), cytosine (C), and thymine (T). Since the discovery of DNA structure by Watson and Crick in 1950s (Watson & Crick, 1953), DNA sequencing has become essential for molecular biology research, and in several applied fields including medical diagnosis, biotechnology, forensic investigation, and microbiology. Up to date, there are about three generations of DNA sequencing technologies from the famous Sanger sequencing to the Next Generation or Next-Gen sequencing technologies such as single-molecule real-time (SMRT) and many others (Heather & Chain, 2016).

The Sanger sequencing is the first-generation technique named after its founder Fredric Sanger in the 1970's. It is a result of the chemical reaction chain-termination of polymerase chains. Sanger sequencing is still a widely used method for sequencing DNA (Shendure *et al*., 2017). This is due to its high sequencing accuracy as compared to the Next-Gen technologies. It is most efficient in sequencing DNA nucleotides with utmost accuracy in short fragments of DNA, small-scale genome projects and also used to validate the results from the Next-Gen DNA sequences.

Sanger sequencing of a genomic region usually involves a chemical polymerase chain reaction (PCR) between a single-stranded DNA template, a sequencing primer, a DNA polymerase, nucleotides (dNTPs), dideoxy nucleotides (ddNTPs), and a pH stability buffer (Goldberg *et al*.,

2006; Pareek *et al*., 2011; Rajeev *et al*., 2015). Next step, the chain-terminated short DNA molecules are detached by their size by means of gel electrophoresis. In gel electrophoresis, DNA samples are filled into one end of a gel matrix, and an electric current is applied; Since DNA is negatively charged, so the short DNA molecules will be drawn towards the positive electrode of the gel. As all DNA fragments have the similar charge per unit of mass, the speed at which the short DNA molecules migrate will be measured by their size. The smaller fragment has less friction to moves through the gel, and hence are will migrate faster than the bigger one (Kircher & Kelso, 2010; Shendure *et al*., 2017). In conclusion, the small DNA molecules will be organized from smallest to largest, whereby reading is of the gel done through a bottom-up approach. The last step simply involves reading the gel to determine the sequence of the input DNA. Because DNA polymerase only synthesizes DNA in the 5' to 3' direction starting at a provided primer, each terminal ddNTP will correspond to a specific nucleotide in the original sequence (e.g., the shortest fragment must terminate at the first nucleotide from the 5' end, the second-shortest fragment must terminate at the second nucleotide from the 5' end, etc.) Therefore, by reading the gel bands from smallest to largest, we can determine the 5' to 3' sequence of the original DNA strand. Sanger sequencing quality relies on the "base calling quality", i.e. the relative certainty with which the base is determined based on the reading by the fluorescence detector in front of which the dye-terminated fragments migrate decreases dramatically towards the end of the fragment, because of the limited separation power of the capillary matrix when it comes to long fragments. The two chain-terminated fragments corresponding to the extension of the two primers are on complementary strands of the template DNA, one of the two fragments have to be reverse-complemented before the extent of the overlap is checked and the two fragments are assembled into one clean sequence. Figure 1 illustrates the Sanger DNA sequencing process from the DNA template, chemical chain reactions and finally the sequencing.

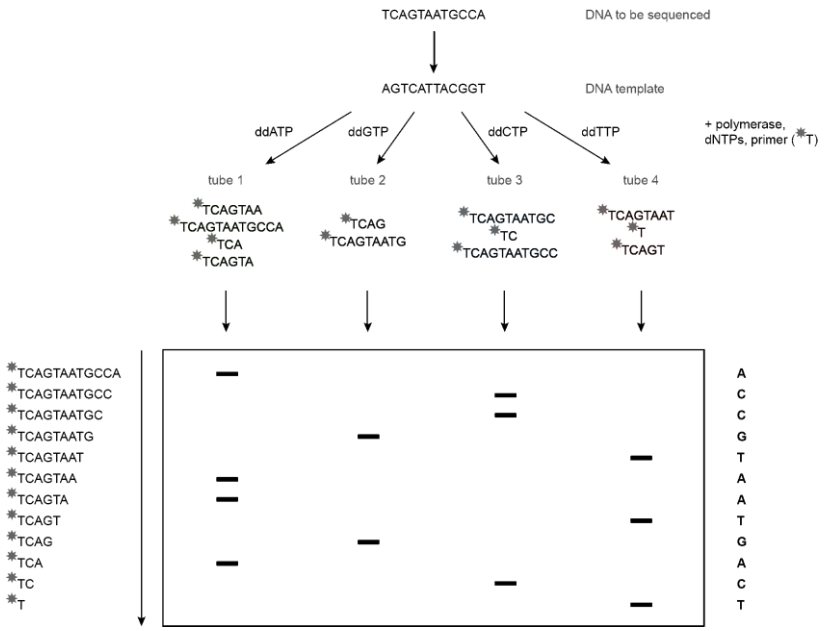


Figure 1: Sanger DNA Sequencing (Sequencing, Forensic Analysis and Genetic)

Through the use of automatic DNA sequencer machines like those of Applied Biosystems, the obtained DNA sequence is then saved as a chromatogram. A chromatogram is a graph showing the result of separating the components of a mixture by chromatography procedure for illustration (Fig. 2). It is saved on a file with an extension *.ab1 or *.AB1. Base-calling is the process of assigning nucleobases to chromatogram peaks. To assess the base calling accuracy of sequenced DNA data, usually, a visual inspection of the sequence trace is done using chromatogram viewing program. Most often proprietary software like CLC Genomics Workbench($5000/year), Qiagen ($4500/year), SeqMan ($5950/year) or the software tools developed by Applied Biosystems sequencer machines are used in big laboratories since they are designed for non-computer professionals and advertised as all-inclusive bioinformatics software (Smith, 2014). Although they attractively designed, powerful, and user-friendly proprietary bioinformatics software tools are expensive for individual researchers, teachers and students to choose the right software for their needs, especially if they do not have a bioinformatics background. Alternatively, there are free software tools for Sanger sequence such as SangerSeqR, Tracy, Phred, ASAP, seqTrace, Shiftdetector (Ewing & Green, 1998; Hill *et al*., 2019; Seroussi *et al*., 2002; Singh & Bhatia, 2016; Stucky, 2012) however, the mentioned tools are limited in usage since most them are command-line based tools, limited to single functionality, and cross platform dependent.

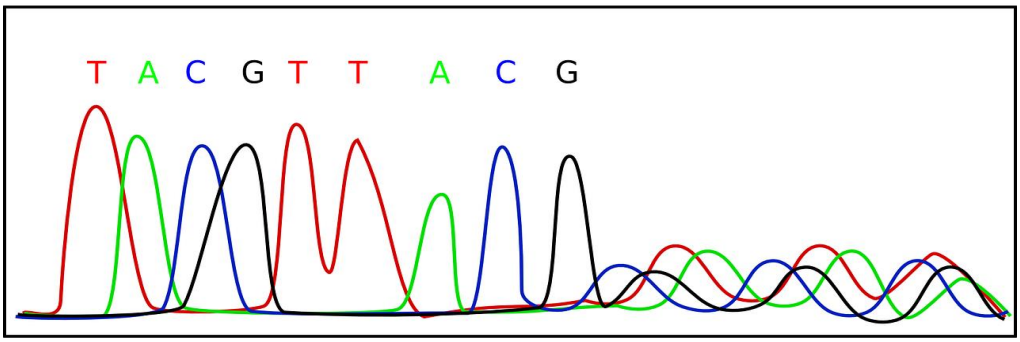


Figure 2: Deoxyribonucleic Acid (DNA) Chromatogram (pixabay.com)

1.2 Statement of the Problem

There are limitations on accessing proprietary software Sanger sequence analysis due to the high costs of the license (Kant, 2010). Alternatively, there have been some attempts to build free, open-source software to deal with the analysis of Sanger sequencing data but also, they are limited since most of them offers specific functionalities. In addition to that, most of these tools are platform-dependent, and some of them are not user friendly since they lack a graphical user interface (Feizi & Wong, 2012).

Hence there is a need to develop a free and open-source tool which will provide a user-friendly graphical user interface, and cross-platform capabilities while providing functionalities such as base-calling, chromatogram viewer, sequence alignment, polymorphism detection and report generation.

1.3 Rationale of the Study

The development of the free and open-source tool will solve the challenges faced by biologists and other researchers, mainly by cutting off the purchasing expensive proprietary licenses for the DNA analysis tools.

1.4 Research Objectives

1.4.1 General Objective

The main objective of this study was to develop a user-friendly cross-platform software tool for analyzing Sanger sequence data at a nucleotide level.

1.4.2 Specific Objectives

(i) To assess the usability of the existing software for Sanger sequence data.

(ii) To develop an automated software for analysis of Sanger sequence data files.

(iii) Usability assessment of developed software.

1.5 Research Questions

To achieve the objectives of the study, the following are research questions:

(i) What is the contribution of the existing software for Sanger sequence DNA analysis?

(ii) How will the proposed software help the analysis of Sanger sequence data files?

(iii) What value is added by the developed software compared to the existing ones?

1.6 Significance of the Study

The study aims to break the barrier for individual biologist or students to accessing the DNA analysis software due to expensive proprietary licenses through the development open-source free tool that will provide proprietary experience. The tool will provide a platform interoperability and user-friendly graphical interface to enable users to focus on the analyses and not learning how to use the system. Moreover, the tool will contribute to the Sanger sequence DNA analysis open-source software community.

1.7 Delineation of the Study

The Sanger sequence DNA analysis will be developed by using the software requirements collected during the review of the existing open-source software tools for Sanger sequence analysis. This is study is limited to the Sanger DNA sequencing method and scoped at the nucleotide level analysis.

CHAPTER TWO

LITERATURE REVIEW

2.1 Deoxyribonucleic Acid (DNA)

Deoxyribonucleic Acid (DNA) is a chemical molecule that contains the living things genetic blueprint (Samanta, Bhaumik, Barman, & Maity, 2017). Deoxyribonucleic acid stores hereditary information that constitutes and maintain living things. The DNA molecule is comprised of four types of nitrogenous bases (nucleotide) namely adenine (A), guanine (G), cytosine (C) and thymine (T). Adenine and guanine are two-carbon nitrogen ring chemical structure known as purines while cytosine and thymine are one-carbon nitrogen ring chemical structure known as pyrimidines. Purines bases bonds with pyrimidines bases to form the nucleotides base pairs which are the building block of the nucleic acids. Figure 3 shows the structure of DNA and the nucleotides chemical structure diagrams.

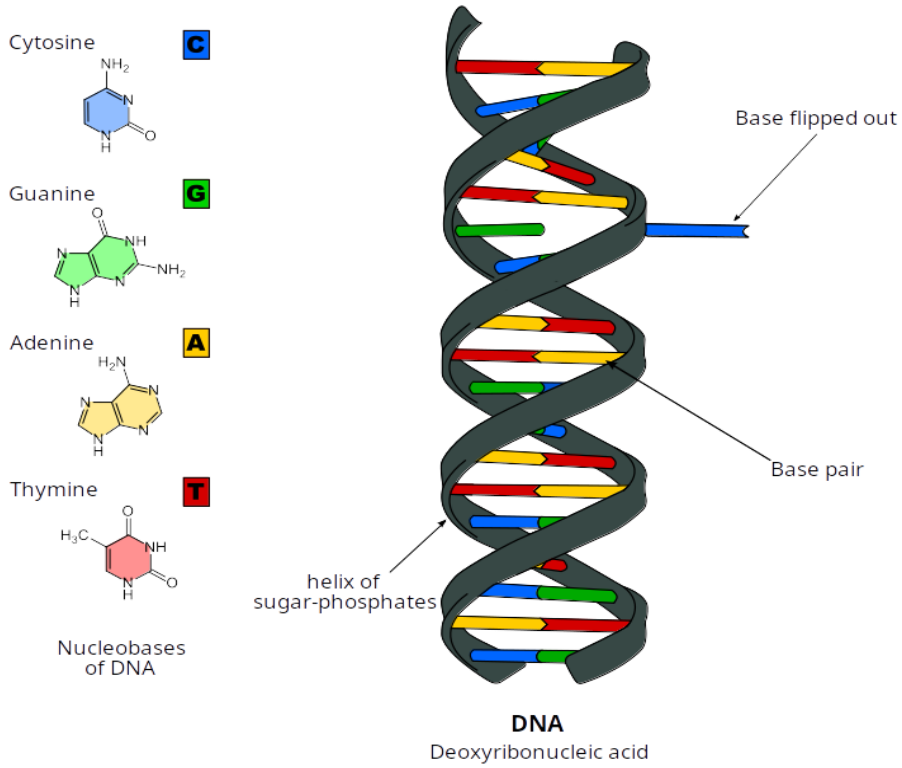


Figure 3: DNA-base (Wikimedia Commons, the free media repository)

The differences in DNA affect how different proteins are created within cells, and determine the biological differences between each living organism (Hingorani, 2013). The DNA sequence variation such as Single Nucleotide Polymorphism (SNP), that occurs when a single nucleotide in the genome varies from other members of a species and may lead to physical change. For

instance, human beings' appearance while others may determine someones' health status (Leaché & Oaks, 2017). Sometimes SNPs do not affect (Leaché & Oaks, 2017), and sometimes they can result in positive or negative effects on the body functions (Clevenger *et al*., 2015; Kuhner *et al*., 2000; Leaché & Oaks, 2017). Different variations in organism genetics can determine such as whether a person is likely to have a food intolerance, how you metabolise different parts of your diet, vitamin deficiencies, and which fitness and skincare regimes are likely to work for a person (Ewens, 2013; Jorde & Wooding, 2004). The knowledge on variations of genes for a particular person enables the decision making in lifestyle changes to optimize ones' health and wellbeing (Gonzaga-Jauregui *et al*., 2012; Zhang *et al*., 2009). Besides, investigating DNA deletion/insertion polymorphism is important since polymorphism may be a result of mutations which frequently cause genetic disease. It was reported that 70% of the mutations in cystic fibrosis patients results from a three base pairs deletion (Luan *et al*., 2013; Martins *et al*., 2019).

2.2 Deoxyribonucleic Acid Sequencing

Deoxyribonucleic Acid sequencing refers to the procedural steps of finding the order of nucleotides in a DNA molecule. It involves techniques and technologies that may be used to determine the order of the four DNA's nucleotides which are adenine, guanine, cytosine, and thymine.

Familiarity with DNA sequences is vital for most biological research and in several interrelated fields such as forensic biology, medical diagnosis, biotechnology and virology (Chmielecki & Meyerson, 2014; Pereira *et al*., 2008). The comparison between normal and varied DNA sequences can help in diseases diagnosis such as various cancers, characterize antibody accumulation, and can be used to guide patient treatment (Abate *et al*., 2013; Chmielecki & Meyerson, 2014; Pekin *et al*., 2011). The rapid methods to sequence DNA would increase more research throughput which is of many advantages especially for precision medical care, and for more living Next-Gen to be identified and registered (Abate *et al*., 2013).

There are basic DNA sequencing techniques like Maxam-Gilbert sequencing which make use of chemical alterations of DNA and subsequent cleavage at specific bases (Maxam & Gilbert, 1977). Also, there is a Sanger sequencing technique which uses the dideoxynucleotides chain-termination approach. Sanger sequencing method when invented used fewer toxic chemicals and lower amounts of radioactivity than the Maxam and Gilbert method. It became the method

of choice, due to its relative ease and reliability (Sanger *et al*., 1977). These techniques are also known as first-generation DNA sequencing techniques.

There are also advanced and De novo DNA sequencing techniques. Advanced techniques perform sequencing of long DNA pieces, such as chromosomes. Also, these sequencing techniques can be used to produce large numbers of DNA short sequences (Delseny *et al*., 2010). On the other hand, the De novo techniques are used to discover novel DNA sequences (Pareek., 2011). Some of these methods are 454 pyrosequencing, Illumina sequencing, Polony sequencing, and SOLiD sequencing. Together these techniques are categorized as high-throughput DNA sequencing also known as Next-Generation Sequencing (Straiton *et al*., 2019).

2.3 Sanger Sequencing

The Sanger DNA sequencing technique involves chain-termination of the elongating inhibitors of DNA polymerase amid the Vitro DNA replication (Sanger *et al*., 1977). This technique is among the most famous methods used for determining sequence nucleotide sequences in DNA (Shendure *et al*., 2017). It is efficient to sequence with utmost accuracy short fragments of DNA (Pareek *et al*., 2011). Sanger sequencing was commercialized by Applied Biosystems where automatic sequencer machines replaced the manual procedures for the DNA sequencing (Biosystems, 2006). Figure 4 illustrates Sanger sequencing processes from the DNA extraction to the generation of the chromatogram. A chromatogram is a visible record such as a graph showing the result of separating the components of a mixture by chromatography procedure in which is stored in the AB1 file.

Sanger sequencing deals with a composite of DNA chains of different lengths, where these fragments are then detached by capillary tube electrophoresis. Electrophoresis is an electro-kinetic method which separates charged particles in a fluid using a field of electrical charge. The method uses an electric field to pull molecules across the capillary fibre (Sanger *et al*., 1977). Additional analysis including both primary and secondary analysis are commonly followed to conclude the analysis. Primary analysis software tools usually come as built-in with the default setting in most of the sequencing platforms (Pereira *et al*., 2008).

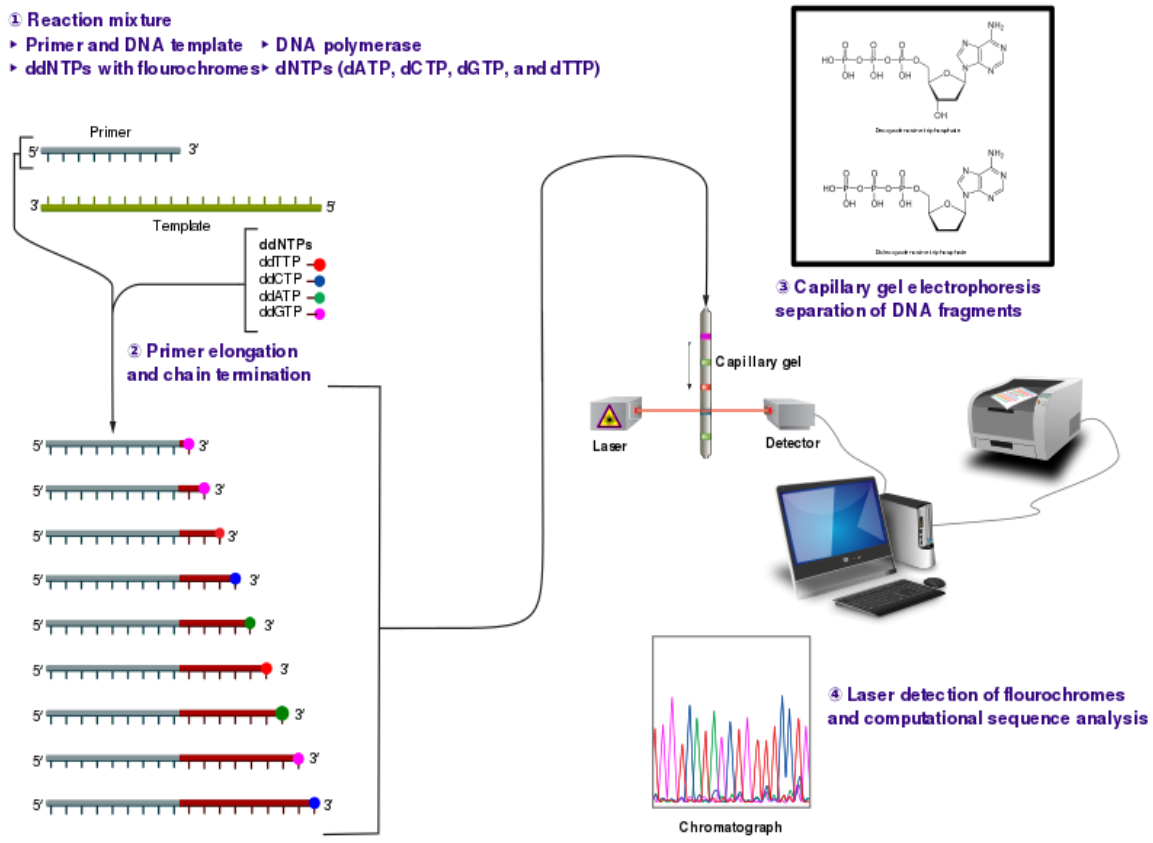


Figure 4: Sanger Sequencing (Wikimedia Commons, the Free Media Repository)

The secondary analysis tools further enhance the sequencing results with features such as polymorphism (the different DNA sequences among individuals, groups, or populations) detection, sequence alignment (a way of positioning the sequences of DNA to identify regions of similarity between the sequences), and production of graphical outputs. Most preferably, software for these tasks are proprietary which require purchasing a license. Most proprietary software are expensive (Loubani *et al*., 2008). On the contrary, free software is available but they are few and some are limited with functionalities.

Raw DNA data contains hidden biological information which are to be revealed for further studies (Rajeev *et al*., 2015). Sanger sequence analysis is vital for the extraction of information of the trace data obtained as a product of Sanger sequencing. Traces obtained need first to be read and assigned bases (nucleotides) on the chromatogram peaks through a base-calling algorithm and later to be visually verified by users through trace viewing program (Lanka *et al.*, 2014). Recently, it has been reported that Sanger sequencing has been substituted by high-throughput sequencing techniques, particularly for large-scale, automated genome analyses (Straiton *et al*., 2019). Nevertheless, the Sanger sequence remains the cornerstone DNA sequencing technique used for small-scale projects, and validation of high-throughput results

(Shendure *et al*., 2017). The technique is more preferred over short-read sequencing technologies due to its capabilities to produce DNA sequence reads of more than 500 nucleotides (Shendure *et al*., 2017).

2.4 Bioinformatics

Bioinformatics is an interdisciplinary field that develops methods and software tools for understanding biological data (Hogeweg, 2011). This interdisciplinary field of science is comprised biology, computer science, information engineering, mathematics and statistics with main goals of analyzing and interpreting biological data.

Bioinformatics uses computational programming as part of the methodology to develop methods, tools and analysis pipelines to study biological data (Singh & Bhatia, 2016). Mainly, bioinformatics is being used to find out the identity of candidate genes and genetic polymorphism (Cornuet *et al*., 2014). These identifications aim to understand better the difference between population, unique adaptation, peculiar properties and genetic basis of diseases (Rozas *et al*., 2017; Singh & Bhatia, 2016). Moreover, bioinformatics is being applied to study the structure and operating principles of nucleic acids and proteins sequences (Pop & Salzberg, 2008).

2.4.1 Bioinformatics Programming Languages

(i) The C / C++

The C/C++ are pre-compiled high-level programming languages that have the best performance in speed of execution since they have wrapping routines and higher-level function calls (Fourment & Gillings, 2008). The downside of these programming languages is memory leaks since they have no garbage collection (Serebryany *et al.*, 2018). They are largely used for building enhanced command-line tool related to aligners and variant-callers (Aruoba & Fernández-Villaverde, 2015). Nevertheless, they still have a steep learning curve as compared to a scripting language such as Python. Most of the programmers tend to avoid C/C++ since it will cost one to write long lines of codes as compared to Python to similar tasks. For this reason, C/C++ are be used for a specific purpose such as to build back-end applications that require fast execution and less memory.

(ii) Perl

Perl was by far the most popular scripting language for handling genetic sequencing data during the ’90s (Fourment & Gillings, 2008). It still has its legacy and there are still many coders who use it as their primary scripting language. It has the Comprehensive Perl Archive Network (CPAN), a software packages repository small install new modules (Tregar & Tregar, 2002). BioPerl, is among the earliest biological repositories that increases the usability of computation methods, for example change of setups to conduct genetic research (Jason *et al*., 2002). There is several biological software that is written in Perl such as GBrowse which is among the popular genome web browser. Excessive test units is still in demand for persons uses, yet it is declining out of use after the rise of Python. Both Perl and Python have comparable accomplish alike tasks and is easier to write code for, especially for new learners (Fourment & Gillings, 2008).

(iii) Python

Python is an open-source interpreted, high-level, general-purpose programming language (Aruoba & Fernández-Villaverde, 2015; Sanner, 1999). Influential, flexible, and easy to use (Aruoba & Fernández-Villaverde, 2015). Python is a simple language for building software tools and applications for data sciences (Boschetti, 2014). It has Biopython, is an open-source repository of uncommercial Python tools and methods for computational biology and bioinformatics, maintained by an international association of developers (Cock *et al*., 2009).

(iv) The R Programming Language

R is a software and statistical programming language that is maintained by supported by the R Foundation for Statistical Computing (Chambers, 2008; R. Gentleman, 2009; Ihaka & Gentleman, 1996). The programming language is free and open-source programming language that produces attractive graphics (Chambers, 2008). Graphs are just great with R programming language. Additionally, the language is broadly used among the mathematical community plus more recently in the data science as well as Artificial Intelligence community (Marwick *et al.,* 2017). It has the central repository (CRAN) which provides easy installation of packages (Hornik *et al.,* 2017). Moreover, it has Bioconductor the biological software repository for R which is the largest repository and community support among others using a different programming language (Gentleman *et al*., 2004; Huber *et al*., 2015). Bioconductor has R-studio packages that are integrated development environment software for usage R in a Matlab style

(Allaire, 2015).

R is an interpreted programming language and it has similar functionalities as those for Python programming language (Aruoba & Fernández-Villaverde, 2015). The difference between R and Python is, R programming language is purely data analysis built-in language while Python is a general purposes programming language (Sugiyama *et al.,* 2018). It is from this ground R is a recent preferably programming language among statistician and especially bioinformaticians since it offers more computational packages for their analyses (Gentleman, 2009; Jiang *et al.,* 2013).

Furthermore, R is flexible and its implementation can be embedded in another programming language such as in C/C++ (Eddelbuettel & François, 2011). This feature makes R computation to be done in cross platforms environments. Also, the execution speed of the program's written in R can be improved by its implementation in fast execution high-level programming such as C and for better memory management (Eddelbuettel & François, 2011). Moreover, R scripts can be written in a way that could establish a connection to the command-based software tools to a front-end application.

2.5 Open-source Software vs Proprietary Software

Open source refers to any things that can be modified and shared since it was designed to be in publicly accessible (Open Source Initiative, 2010). Open-source software is the computer software in which source code is released under a license in which the author grants users the rights to study, alter, enhance and share the software to anyone with no limit of how to use (O'Neill, 2012). This is quite different to the proprietary or closed source software whereby the source code part of the software is hidden for most of the users to access and it can only be manipulated only by the author, team, or an organization created it to change, update, upgrade and fixing faults of the software application (Lee *et al*., 2009). Open-source software is preferred most since they offer unlimited accessibility of the software source codes by anyone with a free license (Open Source Initiative, 2010). This gives control to anyone using the software to decide on how to use the kind of software. Users may modify the software by adding or removing some parts of the application based upon one need (Saldanha, 2004; Zschoch, 2007).

It is important to note that open source software projects offer an eye-opening illustration of new ways to innovate for newbies, scholars and professionals in many fields (Von-Krogh &

Von-Hippel, 2006). Besides, open-source software offers source code reusability whereby similar codes for a particular software can be used in writing another software to perform similar or different tasks, this important for software evolution (Haefliger *et al.,* 2008). Studies also show that open source software is well secured (Herzog, 2016; Hoog, 2011; Payne, 2002; Walden *et al.,* 2009). Open-source software is preferred because of security since it can be accessed and modified by anyone and hence errors or any vulnerabilities might be spotted for correction or fix that might have missed by the program's author (Herzog, 2016). Moreover, it has studied that open source software to be more stable since its source code are publicly distributed (Pirhadi *et al.,* 2016). Recently, users have started gaining trust on that software for critical tasks being sure their tools will not go away if their original developers stop working on them (Gamalielsson & Lundell, 2014). It comes as no surprise that open source software offers big support to users, this being a result of a large community of users and developers (Gamalielsson & Lundell, 2014). Since its software source codes can be accessed publicly it becomes easier to provide support at all levels (Steinmacher *et al.,* 2016).

2.6 Command-line Interface Software Tools

A command-line interface (CLI) refers to the type of a computer program where the user request services from the program through typing successive command lines in form of text (Feizi, 2013). The program has a command-line processor or interpreter to handle the commands (Grant, 2012). In the matter of fact, most operating systems implement a command-line interface where users interactively access the operating system's functions or services. Most popular command-line interpreters include command prompt (CMD) for Microsoft Windows operating system (Sharp, 2007) and bash terminal for Linux and Mac operating systems (Stothard, 2016). Most of the software developers and advance computer users such as system administrators prefer to use CLIs due to its abilities to consume fewer system resources, fast execution and simplified task automation (Barrett *et al.,* 2004). Even though software developers and advanced users rely heavily on CLIs to perform tasks efficiently, most of the normal users rarely use CLIs (Sutter, 2006). The usage of CLIs involves issuing command-lines which are predefined and require to be memorized by users and hence they prefer to use graphical user interface applications (Feizi & Wong, 2012).

2.7 The Graphical User Interface Software

The graphical user interface (GUI) is a kind computer interface that enables users to interact with computer application through graphical icons, menus and pointing devices (Seneviratne, 2008), instead of command-based user interfaces. It is specifically designed to make use of computer's graphics capabilities to make the software application or program much easier to use. GUIs came after the reaction to the perceived steep learning curve of command-line interfaces (CLIs) (Seneviratne, 2008; Shahand *et al*., 2011) which requires commands to be typed in by users to access an application's services. Designing the visual elements and attribute the behaviour of a GUI is a crucial part of human-computer interaction (HCI) software application programming. The graphical user interface aims to enhance the efficiency and ease of use for the fundamental logical design of a stored software application towards satisfaction in a specified environment of use.

2.8 Web Application Technology

Web applications are simply dynamic websites combined with server-side programming to provide functionalities like those provided by a desktop application to interact with users, connecting to back-end databases, and generating results to browsers. Web applications came to solves the software application environment dependencies issues (Nguyen *et al*., 2016). Through a web application, we are now able to share applications across the globe no matter what operating system your electronic device has as long as it has web browsers and an internet connection you can get access to it (Beeley, 2013).

Recently, software development companies rise a strategy to provide web access to software previously distributed as desktop applications (Mikkonen & Taivalsaari, 2011). Since the presentation technology here is entirely changed, it also requires the development of an entirely different browser-based interface application that will adapt the behaviour of an existing application (Guinard, 2011). This technology allows the user to use the application without the need to install it on a local storage disk (Mikkonen & Taivalsaari, 2011).

Development of web applications is much simplified by the use of web application frameworks. Through frameworks, rapid application development has been achieved whereby the development team focus on the parts of their application which are unique to their goals (Ishimaru *et al*., 2014; Vuksanovic & Sudarevic, 2011). Numerous frameworks which are used are open-source software is open source this also has facilitated the development of web

applications since the developers are not limited to framework license (Beeley, 2013).

The use of web application frameworks reduces the number of errors in coding an application, simplifies coding, and reduces development time (Vuksanovic & Sudarevic, 2011). Frameworks can also promote the use of best practices programming so that to overcome security-related problems can be caused by errors in an application developed (Vuksanovic & Sudarevic, 2011).

In bioinformatics also the adoption of the web application development is already happening (Carta *et al*., 2011; Dmitriev & Rakitov, 2008; Romano *et al*., 2007). Web applications become of much help for researchers especially when they are relocated from their research centres. The access of bioinformatics applications via web browser eliminates the tiresome tasks of installing and configuring the application to a computing device and hence gives researchers more time to focus on finding results from biological data.

2.9 Existing Open-Source Software Tools Sanger Sequence Analysis

Base-calling, the procedure of assigning nucleobases to chromatogram peaks and is the basic practice of performing DNA analyses. Phred was among the earliest base-calling software tools which were reported to have less error rate than the ABI machine software (Biosystems, 2006; Ewing & Green, 1998; Machado *et al*., 2011). Phred the command-based tool was the most preferred base-calling tool by both academic and commercial DNA sequencing laboratories. The reason behind its popularity was its high base-calling accuracy (Machado *et al*., 2011). The tool was developed using open-source resources but it is not freely available (Stucky, 2012). It has two types of licenses one for commercial usage which requires purchase and the academic license which requires registration for use and due to this, the tool is categorized semi-proprietary (Scacchi & Jensen, 2012).

Tracy is a free open-source tool for Sanger sequence analyses (Rausch, 2018). It is a command-based tool written in C++ on a Bioconda package. It is a pre-compiled statically linked binary from Tracy's GitHub release page, with a singularity container setup installation file. The tool was developed and maintained by Gear Genomics which is supported by the European Molecular Biology Laboratory (EMBL). Similar to Phred, Tracy was also reported to perform base-call and other tasks such as sequence alignment, assembly and deconvolution of Sanger Chromatogram trace files (Rausch, 2018). Apart from being a powerful tool, Tracy has limitations that it requires C++ programming skills for one to install it for the first time. Also,

it requires external components such as a linker for execution in the command line interface.

SangerseqR was an R implementation for performing analyses of Sanger sequencing data in R (Hill *et al.*, 2019). It was developed as an R package and distributed by Bioconductor on R package repositories (Gentleman *et al*., 2004; Huber *et al*., 2015). The package contains libraries for reading AB1 files, performing base-calls and plotting chromatograms. Among the limitations of SangerseqR is the literacy of the R programming language. Usage of this package and its tools requires R programming language commands to perform the above-mentioned tasks which on the other hand may be cumbersome for users with limited or no experience with R programming language (Leipzig, 2017).

Automated Sanger Analysis Pipeline (ASAP) is a tool for rapid analyses for Sanger sequence data (Singh & Bhatia, 2016). It is one among other reported command-based tools for Sanger sequence analyses. ASAP performs similar tasks to those of SangerseqR such as reading AB1 files and extracting raw data and alignment but it is automated (Singh & Bhatia, 2016). The developer of ASAP attempted to automate some external programs EMBOSS, NCBI BLAST+, Seqtk, Python 3.x, BioPython to perform tasks. The tool is more complex for users with little or command-line interface interaction (Leipzig, 2017; Seemann, 2013). In addition to that the external programs require individual installation on the host computer. If one misses installing even one program, it may lead to malfunctioning of the tool (Seemann, 2013).

SeqTrace is a graphical user interface tool for Sanger sequence analyses which can be obtained as free and open-source software (Stucky, 2012). It performs tasks similar to the above-listed tools with the addition of batch processing. SeqTrace was developed with built-in dependencies to make the application free from the installation of additional bioinformatics software packages (Stucky, 2012). SeqTrace can identify, align, and compute consensus sequences, but it lacks variants detection functionalities which are essential for nucleotides analyses.

Developers made attempts to compute for polymorphic and superimposed trace files to estimate the number of insertion/deletion mutation results on the web applications tools that include ShiftDetector, Indelligent, CHILD, and Mixed Sequence Reader (Chang *et al.*, 2012; Dmitriev *et al.*, 2008; Seroussi *et al.*, 2002; Zhidkov *et al.*, 2011). ShiftDetector was developed in a window-based comparison with capabilities to report cases of shift mutation and predict the sequence using statistical methods to initiate the shift (Seroussi *et al.*, 2002). However, its limitations include not accounting for the multiple hypothesis tests involved in considering

many windows and many possible gap sizes, and also for not accounting for sequence composition biases (Zhidkov *et al*., 2011). Indelligent on the other side employed dynamic programming optimization to predict superimposed allelic sequences solely from a string of letters representing peaks within an individual mixed trace which also reported to have limitation in providing statistical tests (Dmitriev & Rakitov, 2008; Zhidkov *et al*., 2011). CHILD was designed with statistics in mind and the implementation of FASTA alignment algorithms to detect bundles of insertions and deletions in DNA sequence traces (Zhidkov *et al*., 2011). Comparably, CHILD was more accurate and sensitive in detecting rare variants than Indelligent and ShiftDetector, however, CHILD web application is currently unavailable (Zhidkov *et al*., 2011).

The literature indicates the availability of some open-source Sanger sequence analysis tools (Hill, 2015; Singh & Bhatia, 2016; Stucky, 2012). Most of the tools are command-based tools Phred, SangerseqR, ASAP and some are GUI based desktop tool such as SeqTrace which limited to Mac and Linux operating systems. Web application tools are also available including ShiftDetector, Indelligent, and CHILD but they are limited to only a specific functionality, polymorphism detection. In addition to that, some studies reported some tools that are open-source but not freely available for instance Phred, also some of the tools are open-source but are not available for example CHILD tool. It may be drawn out this summarization that most of the existing tools for Sanger sequence analysis are command-line based, specific task-oriented, and platform-dependent and as a result leads to limited usage (Jackson *et al*., 2011). It is from the grounds comes a need to have a tool will first being free and open-source that will have multitasked functionalities, platform interoperability and a user-friendly graphical interface.

2.10 Usability

Usability simply refers to how easily can a specific user of a specific product can use the product or design to accomplish intended goals effectively, efficiently and acceptably. Usability also in the field of human-centred interaction is defined as a way to remove all possible frustration that users user may experience when using a product or design (Wilson, 2013). On the other hand usability evaluation refers to a method used in the centered design which is used to assess a product or design by testing it with a group representatives users (Andrews *et al*., 2012; Koziokas *et al*., 2017). Moreover, it is a platform for users give direct feedback and recommendation on how the feel and find about the product (Bergstrom &

Schall, 2014). Usability was founded as a results basic quality component which are Learnability, Efficiency, Memorability, Errors Tolerance and Satisfaction (Nagaraj *et al*., 2014). It is a good practice for the product or design be made for production to be tested by real users and get their insights (Wilson, 2013).

2.10.1 Usability Evaluation Methods

The usability evaluation method refers techniques that are used for collecting data during test session from users when are interacting with a product with aim achieving usability (Fernandez *et al*., 2011). Studies revealed several usability methods for testing products and among them usability testing and heuristic evaluation have been the most appropriate methods (Davis & Jiang, 2016; Fernandez *et al*., 2011; Quiñones *et al*., 2018). Heuristic evaluation most of time done by professionals who use the generally accepted guideline to evaluate the usability of the product through demos and report issues. In contrast, usability testing recruits users to evaluate particular product usability through their feedback after interacting with the product (Fernandez *et al*., 2011).

CHAPTER THREE

MATERIALS AND METHODS

3.1 Study Area

This study took place at the molecular biology dry laboratories at the Biosciences east and central Africa - International Livestock Research Institute (BecA-ILRI) research hub in Nairobi, Kenya. At BecA-ILRI hub there is access to high-speed computational software tools for research and mentorship in Bioinformatics. At BecA-ILRI hub, the work focused on study design mainly in understanding the existing Sanger sequence analysis software tools and determining the requirements for the development of the user-friendly open source software tool. The development and validation of Sanger Sequence Automatic Analysis Software Tool (SSAAT) was done at NM-AIST CoCSE laboratory.

3.2 Scope of the Study

The study aimed at development of a software tool for DNA analyses at the nucleotide level. The aim of choosing this scope was to build a software tool which could perform basic analyses of which later could be scaled up for more advanced tasks.

3.3 Methodology

The study was done through Agile methodology. This kind of approach was selected in order to enable users interaction and fit their requirements during development (Bisandu, 2019; Martin & Euchner, 2012). The Agile methodology elevates continuous iteration of software development and testing throughout the project life cycle (Bisandu, 2019). The development and testing activities were concurrently conducted to meet user requirements (Dorst, 2011).

This study applied Lean software development life cycle which is an iterative Agile methodology. The Lean software development utilizes elements of manufacturing processes such as eliminating waste, amplify learning, late decisions, and reduced delivery time (Ebert *et al*., 2012; Pernstål *et al*., 2013; Poppendieck & Cusumano, 2012). The methodology focuses on delivering value to the users through effective value stream mapping. Moreover, the methodology is evolving, flexible and with no rigid guidelines or rules (Poppendieck & Cusumano, 2012).

The development of the software tool started with concept development as the initial stage of process where scope and tasks prioritization are defined. In this phase project goals and outcomes were outlined and plans to accomplish them were set through understanding of the main problem and getting a big picture of the solution. This involved seeking advice from experts to find out more about the area of concern through observing, engaging end-users and understanding of the existing issues.

Information gathered during the initial stage, was being processed for further action during inception stage. Analyses and synthesis of observations was done in order to determine the core problems. The identification and definition of the problem was done in a human-centered manner so as to capture user requirements at a wide angle. It was found from the existing software tools that there was a need to have a user-friendly software tool that would have a graphical user interface, with cross-platform capabilities to work in most the operating environments.

3.4 Requirement Engineering

Requirements engineering is a field of expertise that is applicable in various situations and processes (Curcio *et al*., 2018; Sillitti & Succi, 2005). It is useful even in agile software development and can help to provide more substance to the Lean development framework (Curcio *et al*., 2018).

In Lean, the software backlog is a dynamic set of requirements, with the software stakeholders responsible for its content and management. This should be done continuously and in collaboration with both stakeholders and the developer so as to meet expected results (Curcio *et al*., 2018).

3.4.1 Requirement's Specification

Formal software requirement models are being produced during this activity. Requirement's specification refers to the detailed description of a software system to be developed. This involves capturing and maintaining the systems' functional and non-functional requirements. The functional requirements are the one which describes what services the system will provide. On the other hand, non-function requirements are the one which specifies criteria that can be used to judge operation of the system instead of specific behavior. The non-functional requirements specify the system's quality characteristics or quality attributes.

Table 1: Functional Requirements

Functional Requirement ID	Functional Requirements	Descriptions
FR001	Upload DNA sequence data files	The system should enable users to upload the DNA sequence (AB1 files) for the DNA sequence analysis.
FR002	Perform Base-calling	The system will automatically perform base-calling by assigning nucleotides of the uploaded DNA sequence data to chromatogram peaks.
FR003	Display chromatogram	The system should enable users to view the chromatogram.
FR004	Chromatogram settings	The system should provide chromatogram option for the user to set how to view the chromatogram.
FR005	Convert AB1 file to FASTA file format	The system should automatically convert AB1 to FASTA file format of the DNA sequence file.
FR006	Extract DNA sequence file details	The system should extract DNA sequence details from the uploaded AB1 files.
FR007	Sequence alignment	The system should perform DNA sequence alignments and display results.
FR008	Polymorphism detection	The system should perform SNP and InDel polymorphism detection of the DNA sequence and display results.
FR009	Report generation	The system should be able to generate report and enable users to select and download results.
FR010	Operational manual/instruction	The system should provide to users the system operating manual/instructions.

Table 2: Non-Functional Requirements

Non-Functional Requirement ID	Non-Functional Requirements	Descriptions
NFR001	Usability	The system should be easy to use, effective, efficient and satisfactory in achieving quantified objectives.
NFR002	Reliability	The system should have minimum downtime error during operation.
NFR003	Performance	The system should have high throughput with short response time.
NFR004	Availability	System should be available with high mission capable rate.

3.5 Software Tool Design

The designing of SSAAT was done at BecA-ILRI hub by five participants with different backgrounds as shown in the Table 3 below. All participants were involved in the designing of the developed tool under the supervision of the senior scientist. The molecular biologist and the bioinformaticians since were the main stakeholders of the tool, hence also had an important role to check if the software requirements were met through the testing of the developed prototype. Moreover, the software engineer had three roles to design, develop and to test the tool prototype.

Table 3: Sanger Sequence Automatic Analysis Software Tool Designing Participants

No.	Field of expertise	Level of expertise	Degree level	Roles
1	Bioinformatics	Senior scientist	PhD	Supervising
1	Bioinformatics	Scientist	Masters	Designing and testing
2	Molecular biologist	Scientist	Masters	Designing and testing
1	Software engineering	Software engineer	Masters' student	Designing, developing and testing

3.5.1 Design Concept

Sanger Sequence Automatic Analysis Software Tool was designed in an Agile way, whereby the existing software tools were identified and explored using empathic approach through user verification. It was found that the existing Sanger sequence software tools had lack of user

interface which also led to difficulties of use (Feizi & Wong, 2012). Also the software tools had difficult installation, in order to the software tools could require a command-line literacy (Feizi & Wong, 2012). Moreover, the existing free open-source software tools had lacked the cross-platform interoperability capabilities and hence worked only to selected operating systems.

From the above mentioned it was suggested that the new software tool to be developed as a web application so as to address the challenges previous faced in the existing software tools. The web application would have a graphical user interface that can be accessed by users through any device that have web browser. This would have solved the challenges for graphical user interface, installation of software to user's devices and also interoperability since it is accessed through a web browser (Burzacca & Paternò, 2013; Nguyen *et al*., 2016).

During the tool design wireframe also known as screen blueprint was used to give a visual guide that represents the skeletal framework of a web application to be developed. Figure 5 depicts the design of the proposed DNA analysis tool to be developed through a wireframe diagram.

The wireframe has six components as it represented through labels from 1 to 6 in Fig. 5. Label 1 represents a sidebar that holds tool settings component which offers options for user to upload data chromatogram viewer, and sequence alignment settings. Label 2 represents DNA sequence component where users will be able to get the DNA sequence details. Additionally, the DNA sequence alignment and polymorphism detection components were represented by label 3 and 4 respectively. Lastly, report generation component was represented by label 5 while the tab 6 represented the chromatogram viewer. The prototype of the tool was developed as it.

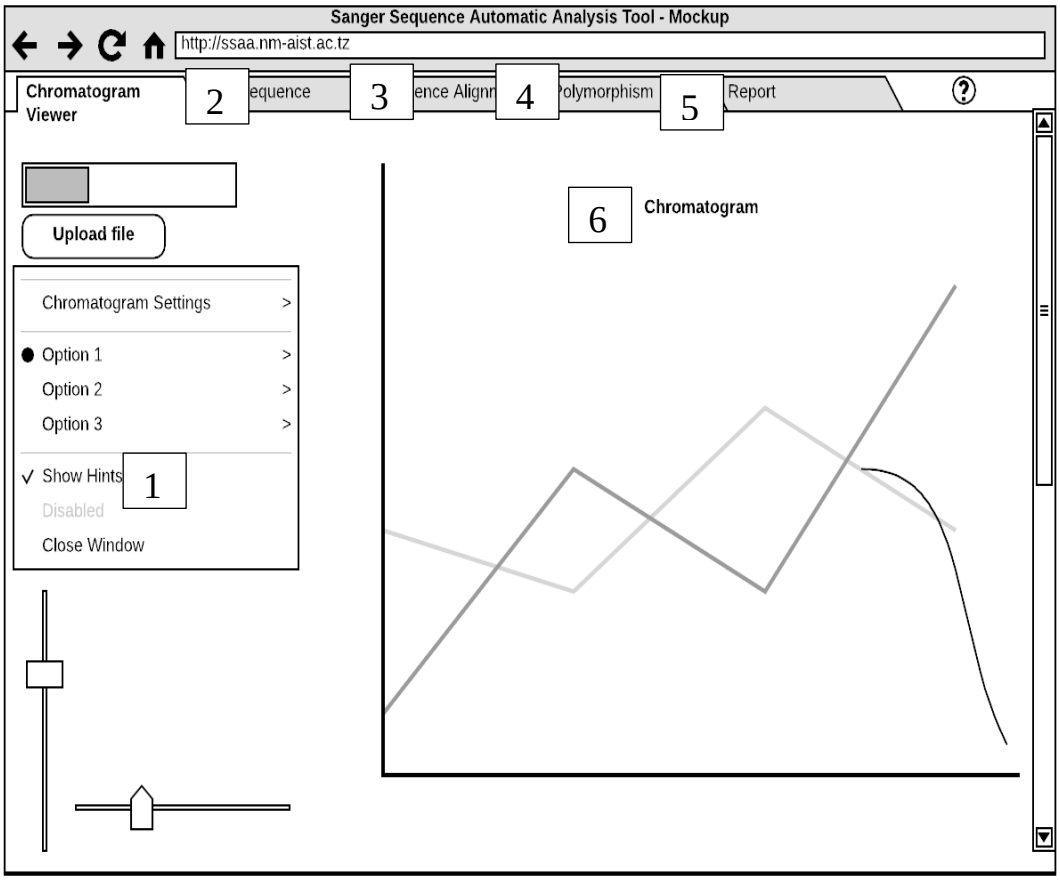


Figure 5: The SSAAT Design Wireframe

3.5.2 Unified Modeling Language (UML) Use-Case Diagram

Use case diagrams are the Unified Modeling Language (UML)'s behavior high level diagrams that is comprised of Systems, Actors, Use Cases, and Relationships. They are simple models used to schematically document the functions of a system from a user's perspective and to represent the interrelations of the functions of a system. Use case diagrams were used to document the system's behavior in a high abstract level for the purpose of simplifying the illustration of the proposed software tool to the stake holder. The UML Use case diagrams is presented in Fig. 6 where the rectangle helps define the scope of this system and anything within this rectangle happens within the software tool while on the other side anything outside this rectangle does not happen in the tool. The next element is an actor, which is depicted by this stick figure. An actor is going to be someone or something that uses our system to achieve a goal. That could be a person, an organization, another system, or an external device. In this software tool the main stake holder is the researcher and hence becomes an actor to the system. Lastly, a Use Case is depicted with this oval shape and it represents an action that accomplishes some sort of task within the system. They are placed within the rectangle because they're

actions that occur within the software tool SSAAT.

3.5.3 Unified Modeling Language Sequence Diagram

Sequence diagrams are a type of UML diagram that show how objects in a system or classes within code interact with each other. Particularly these diagrams show interactions in the order they take place, in other words, they show the sequence of events. Sequence diagrams are primarily used by developers and business professionals to document processes or understand the requirements of a new software. In this study, sequence diagrams were used to emphasizes how objects communicate and the time ordering of the messages between objects. The sequence diagram of SSAAT is illustrated in Fig. 7, whereby the objects that participated are placed on top. The objects that initiate the interaction are at the left, and place increasingly more subordinate objects to the right. So basically, this reflects the way the events will flow for the majority of the interactions in the system.

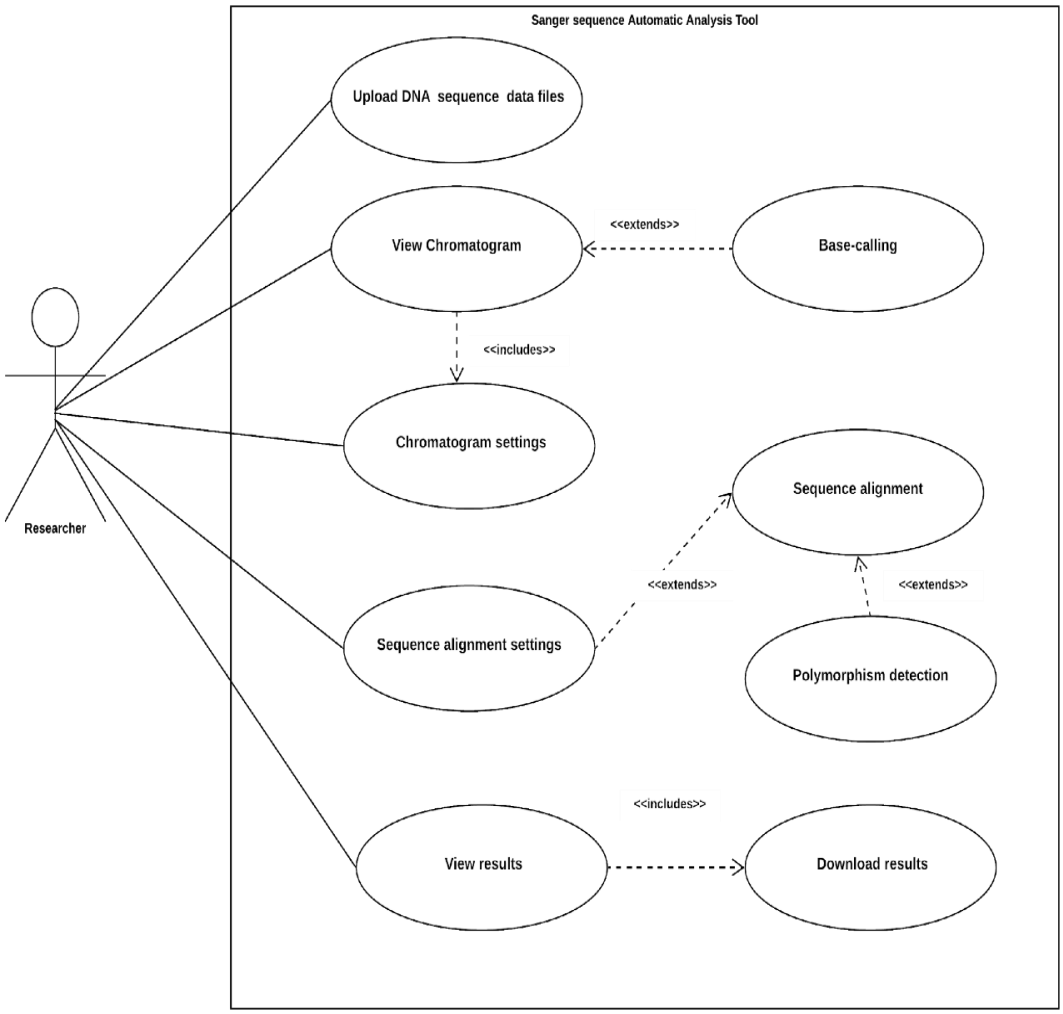


Figure 6: UML Use Case Diagram for Sanger Sequence Automatic Analysis Software Tool

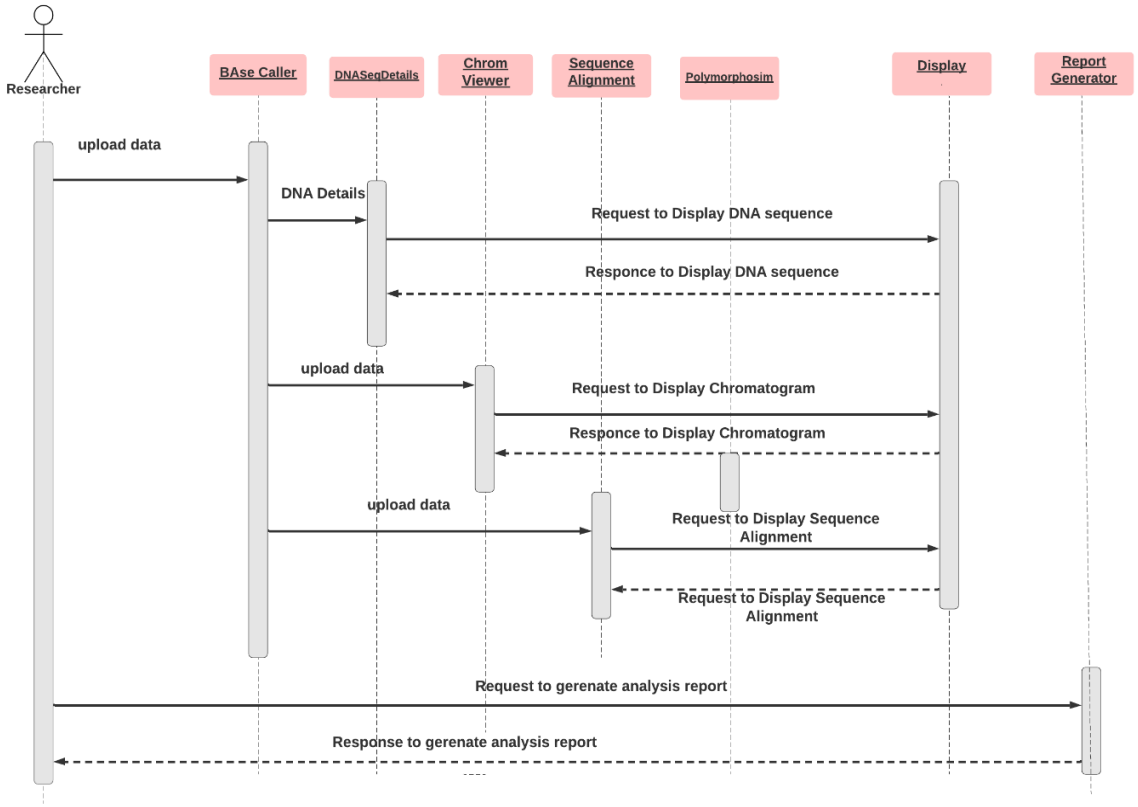


Figure 7: UML Sequence Diagram for Sanger Sequence Automatic Analysis Software Tool

3.5.4 Architectural Design

The SSAAT was developed based on the client-server architecture. This is a distributed computing architecture that offers the server which delivers and manages most of the resources and services to be consumed by the user known as client (Oluwatosin, 2014). Mostly, clients and servers communicate over a computer network on separate hardware, but in some cases both of them may reside in the same system (Reese, 2000). The server host runs one or more service applications which share their resources with the clients. On the other hand, a client does not share any of its resources, but requests service from a server. Clients are the communication session's initiator with servers since they do request service and wait for response from the server. The client side is the part that enables system interaction with users through the graphical user interface which may be accessed through web browser. Client side of SSAAT provides manipulation mechanism of the input and output of data for analysis.

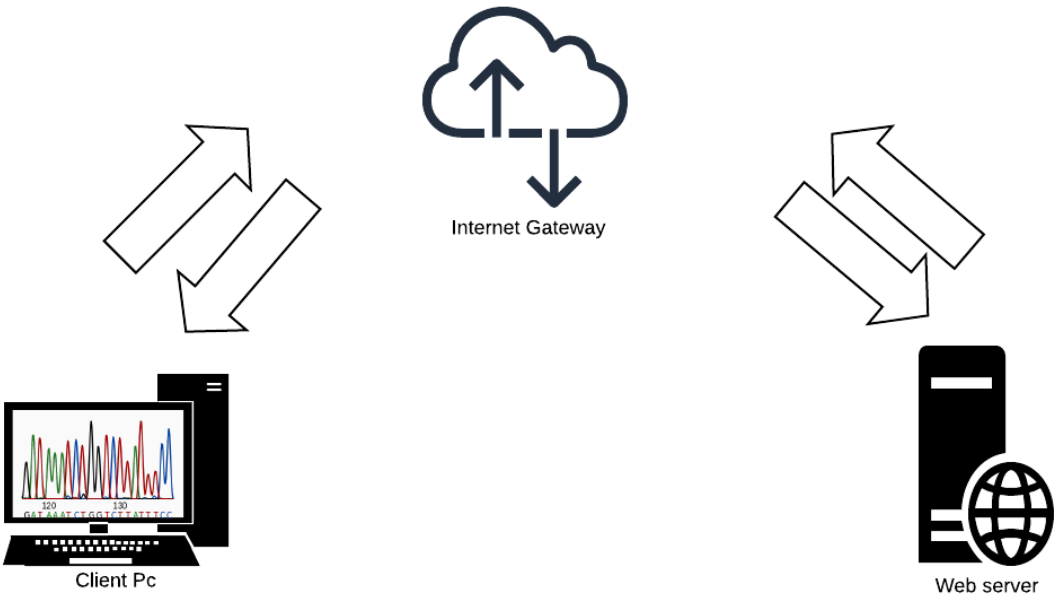


Figure 8: Sanger Sequence Automatic Analysis Software Tool Architectural Design

3.5.5 Data Flow Diagram

Data flow diagrams (DFD) are graphical representation of the flow of data through an information system. Data flow diagrams tell how the data travels within an application so basically, during software development it is most important to understand how your data will travel from one module to another within your application. Figure 9 illustrates the DFD as data flows from the AB1 file reader where the DNA files is uploaded, then the processed data flows to the base-calling algorithm which here it acts as the central processing unit of the tool. All other processes depend on the base-calling process to assign the nucleotides in a well-defined order for them to perform respective tasks such as the visualization of chromatogram, DNA sequence details, sequence alignment and the rest. Data flow diagrams main purpose is to ensure that the software is well organized as the same time acts as a checklist to confirm that the software will have all the specifications are met through a visual aid.

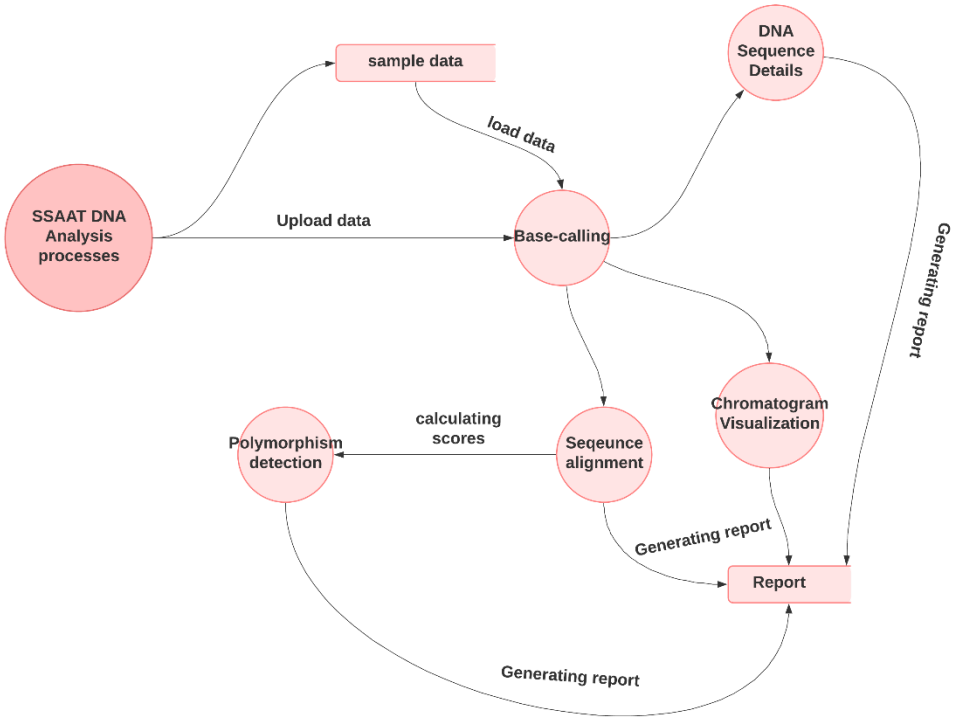


Figure 9: Data Flow Diagram

3.6 Development

The development of SSAAT required an approach that could facilitate making of a tangible software tool out of design concept at early stages to be tested by users. Testing of the software tool at early stage provides a room to refine and validate software designs that will lead to a right product.

3.6.1 Prototype Development

Development in software engineering refers to planned and structured activities performed to transform users' requirements into software products. This involves capturing user requirements, designing, development by coding and maintaining the source code of a functional software. The development approach used in this study was Lean software development life cycle. Evolution prototyping was used so as to eliminate waste and facilitate quick development (Poppendieck & Cusumano, 2012). The scaled down version also known as prototype of the software tools were developed so that they could be used to investigate the problem solutions generated in the previous stage. This was an experimental phase, and the aim was to identify the best possible solution for each of the problems identified during the first three stages.

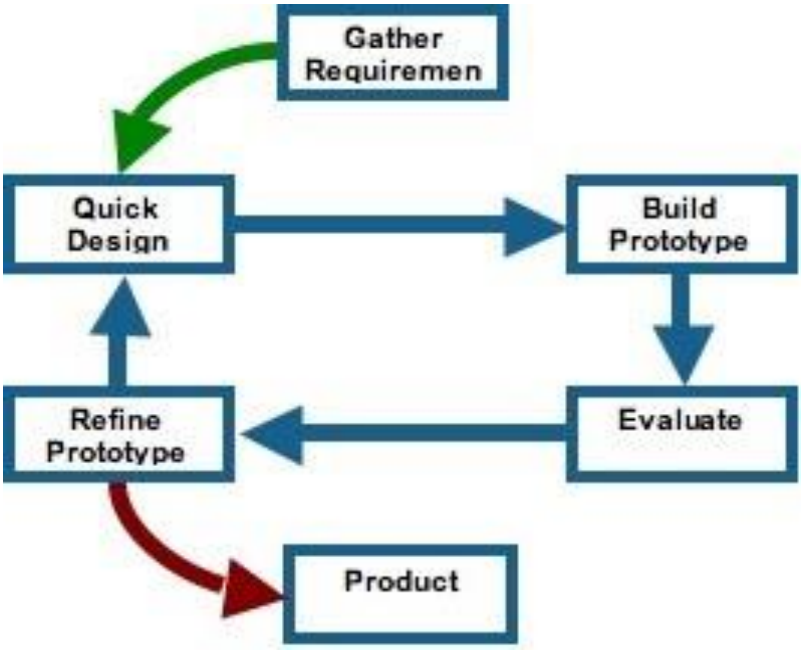


Figure 10: Prototyping Software Development Life Cycle

It is through prototype development that problem solutions are implemented and examined for approval either accepted, improved and re-examined, or rejected on the basis of the users' experiences (Hillgren *et al*., 2011). At the end of this stage, the better idea of the constraints inherent to the software tool and the problems that are present, and were clearly observed (Hillgren *et al*., 2011).

In this study the web application software tool prototype was developed using the R-Shiny framework. The prototype was refined iteratively for improvement. The prototype was then evaluated and improved performance and design in every stage of iteration until the final prototype was obtained. Mock-ups were used to imitate the feel of the design during development of the prototype as it can be illustrated on Fig.11.

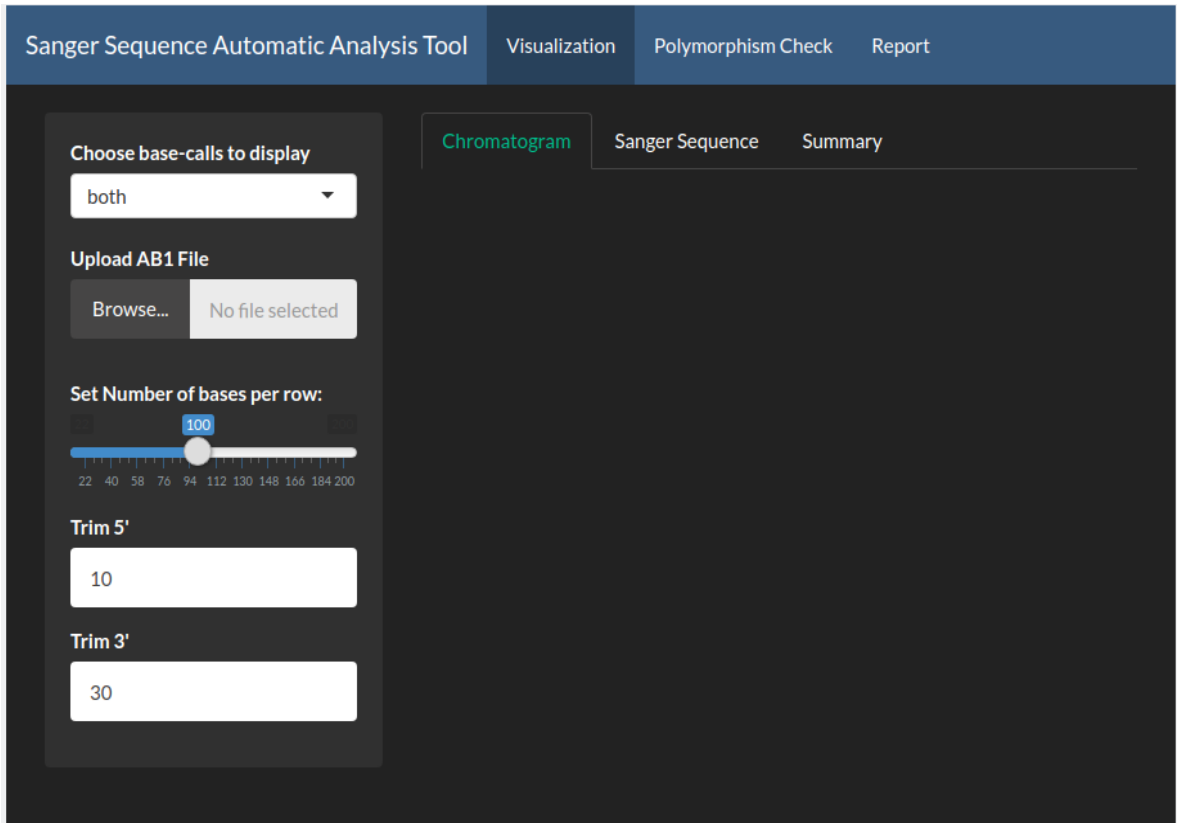


Figure 11: Sanger Sequence Automatic Analysis Tool Mockup

(i) Shiny Web Application Framework

In this study a web application for Sanger sequence analysis was developed using Shiny web framework. The R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing (R Development Core Team, 2011). The R language is widely used among statisticians and data miners for developing statistical software and data analysis (Chambers, 2008). R is well known for solving analytic challenges (Chambers, 2008).

Recently, most of web application development projects employs the use framework for development rather than developing from scratch (Kumar & Kumar, 2019). The reason is because frameworks have standard reusable components and API that can implemented to develop new web application (Vuksanovic & Sudarevic, 2011). This ease testing activities and also reduces time for development (Kumar & Kumar, 2019; Vuksanovic & Sudarevic, 2011). Shiny web application framework was selected for the development of Sanger sequence analysis application because of its capability for developing interactive web applications for data analysis and visualization (Beeley, 2013). The framework based on R programming language, HTML, CSS and also supports JavaScript scripts. Once a Shiny application is built it becomes an R interface that anything could be done with R, could be done also in the

application. Moreover, Shiny supports prototyping which was more suitable for the development of Sanger sequence analysis software tool since prototyping methodology was used.

(ii) Hyper Text Markup Language (HTML) and Cascading Style Sheet (CSS)

Development of the web application page used Hyper Text Markup Language (HTML) and Cascading Style Sheet (CSS) in creating web pages (Gabarro, 2015). Hyper Text Markup Language is used in web technology for access and display web pages in the web browser. It describes the structure and appearance of the web page. The HTML works together with the CSS which is used to describing the presentation of the web page. The used HTML and CSS for development of graphical user interface by creating web pages to interface the application for easy user interaction.

(iii) RStudio

RStudio is an open source Integrated Development Environment (IDE) for R programming language (Allaire, 2015). It provides an environment with built-in libraries and interpreter to write, run and test R programs and applications. In this study RStudio was used for write R codes, running and testing the developed web application for Sanger sequence analysis (Allaire, 2015).

(iv) Other Requirements

- Internet connection
- Web application server
- Sanger sequence DNA files.

3.6.2 Assumptions

The development of the Sanger sequence analysis software tool was based on several assumptions for the software and hardware features. During the development of the web applications the following were assumptions about the client and server environments:

(i) The server machine system that host the server application will have the required resources such as Linux operating system, 8 GB memory, 120 GB storage space, 3.0 GHz central processing unit (CPU) and above to run the web application.

(ii) A web browser is available in the client computer where the web application will run.

(iii) Both the client and server machine will have TCP/IP network connectivity for communication in order to run the web application.

(iv) Users have good knowledge of computer and able to use web browser to access information from the world wide web through internet connection.

3.6.3 Software Tool Components

The development of SSAAT web application was done component wise, whereby each component was developed individually and later combined to work together. Figure 12 below shows the building components of SSAAT starting with the first on the AB1 file reader up to the last, the report generator.

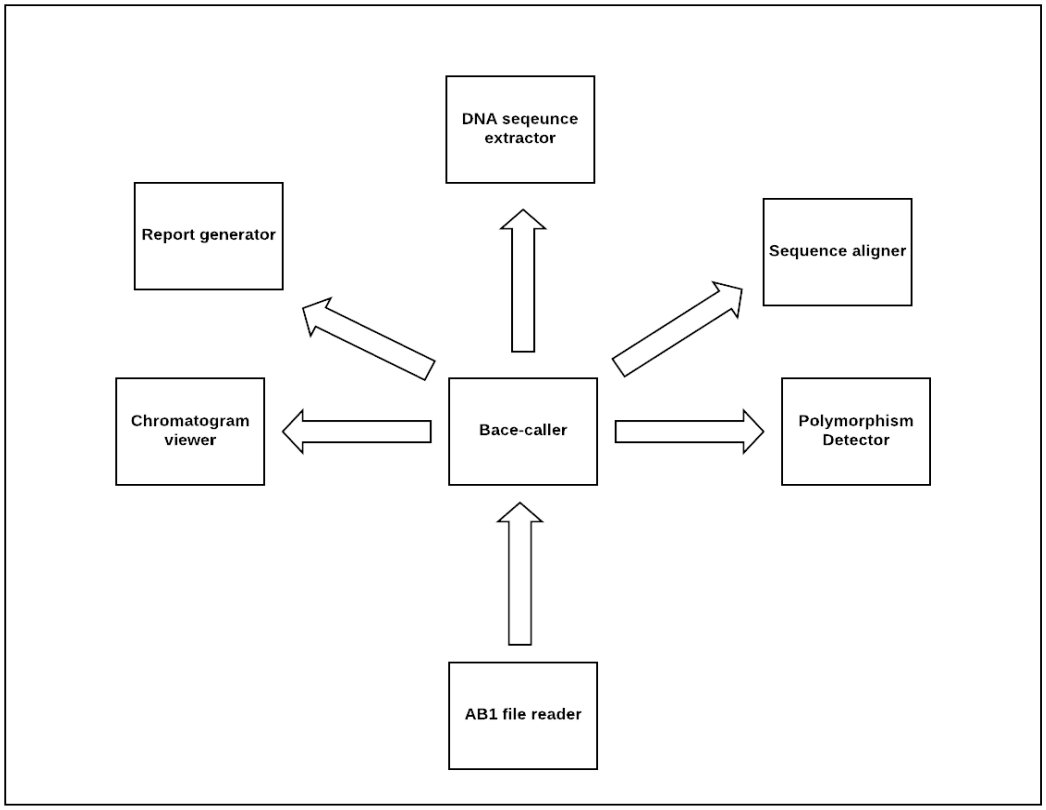


Figure 12: Sanger Sequence Automatic Analysis Software Tool Components Block Diagram

(i) AB1 Files Reader

The AB1 file are one that stores the Sanger sequence DNA as a raw data. In order to extract the raw DNA data, one requires and software tool to read the AB1 file. The AB1 file reader for SSAAT was developed by implementing an R package SangerseqR (Hill *et al*., 2019) found at Bioconductor repository (Huber *et al*., 2015). The package provides libraries to read AB1 files and extracting the raw Sanger sequence DNA data for further analyses such as base-calling and others. The AB1 reader was with a graphical user interface for file browse and upload field. This field was developed so as to enable user to easily select and upload the AB1 file for analyses.

(ii) Base-Caller

Base calling is the process of decoding the output signals of the Sanger sequence reactions into sequence reads (Sheikh & Erlich, 2012). Base-caller program for SSAAT was developed by implementing SangerseqR base-calling libraries (Hill *et al*., 2019). The developed base-caller used Phred base-calling quality score the one found at the widely used base-calling software Phred (Brent-Ewing & Green, 1998). A Phred quality score is a measure of the quality of the identification of the nucleotides generated by automated DNA sequencing (Roberts, 2011). Phred quality score Q are defined as a property which is logarithmically related to the base-calling error probabilities P.

$$Q = -10log_{10}P$$

OR

$$P = 10^{\frac{-Q}{10}}$$

Where P = Base-calling error probabilities and Q = Phred quality score in Decibel (dB).

Table 4 below shows the relationship between Phred quality score and base-call accuracy. It can be clearly observed that, for instance if the base-caller program assigns a quality score of 40 to a base, the chances that this base is called incorrectly are 1 in 10 000 which is equivalent to 99.99% accuracy. The quality scores are assigned to each nucleotide base call in automated sequencer traces (Roberts, 2011).

Table 4: Phred Quality Score and Base-Call Accuracy Relationship

Phred Quality Score	Probability of incorrect base call	Base call accuracy
60	1 in 1000000	99.9999%
50	1 in 100000	99.999%
40	1 in 10000	99.99%
30	1 in 1000	99.9%
20	1 in 100	99%
10	1 in 10	90%

(iii) Deoxyribonucleic Acid Sequence Extractor

Deoxyribonucleic Acid sequence refers to the order of nucleotides of a particular DNA sample. The DNA sequence extractor of SSAAT was developed as an R program to fetch data from base-caller program. Data that could be fetched by DNA sequence extractor program includes DNA sequence, AB1 file details (Schema, 2009) and base-call quality score of data.

(iv) Chromatogram Viewer

Chromatogram viewer was developed for visualization of the chromatogram trace file generated after base-calling. The chromatogram trace file contains visible records in form graphs that shows the result of separating nucleotides mixture. The chromatogram viewer was developed to provide an interface for users to visualize and inspect the DNA sequence chromatograms. Moreover, chromatogram viewer was developed with a built-in single nucleotide polymorphism (SNP) detector and highlighter.

(v) Deoxyribonucleic Acid Sequence Aligner

Deoxyribonucleic Acid sequence alignment refers technique of positioning the sequences of DNA in order to identify areas of resemblance that may be an outcome of functional, structural, or evolutionary relationships between the sequences (Greene, 2016). The SSAAT sequence aligner was developed by implementing a Biostring (Pagès *et al*., 2017) package from Bioconductor repository (Huber *et al*., 2015). Biostring contains libraries for performing DNA sequence alignment with five different alignment setting local, global, overlap, global-local and local-global alignments which were all implemented. Also, the DNA sequence aligner was developed with a user interface field for users to upload a reference sequence for alignment

with DNA sample. The field developed in way accept .txt or. Fast-All File Format (FASTA) file format since most of reference sequence are obtained in the format mentioned before.

(vi) Polymorphism Detector

The DNA polymorphism refers to differences of sequence as compared to a standard reference that is present in at least 1–2% of a population. The SSAAT polymorphism detector was developed to detect polymorphism of DNA sample by implementing Smith-Waterman algorithm from a . The algorithm uses dynamic programming to align two sequences in a more quantitative way by giving scores for matches and mismatches through scoring matrices.

(vii) Report Generator

Sanger Sequence Automatic Analysis Software Tool was developed with a program for report generation. The report generator was developed to capture analysis information including the DNA sequence details, chromatogram trace file, sequence alignment and polymorphism details. The programs provide PDF and Microsoft Word file format options for users to download the generated report.

3.7 Testing

The SSAAT web application was tested during development using unit testing, integration testing, system testing and usability testing to validate the developed web application. Unit testing was done as the first level of software testing where individual components of a software tool were tested to check if components developed meets the requirements.

Integrated testing was done after combining all the individual software components of the software tool. The bottom-up approach was used to perform integration testing where bottom level units are tested first and upper-level units' step by step after that. In this task gray-box testing method was applied to test the interfaces of the integrated components.

System testing, performance and compatibility testing was done at the end of development whereby the web application was tested at both at the localhost and remote the Shiny web server host at the NM-AIST web server. Google PageSpeed Insights and Apache JMeter were used to test device compatibility and performance of execution at the real working environment (Apache, 2014; Google, 2015).

3.8 Validation of SSAAT

Validation of the tool was done through the usability testing which the specific object number three of this study. The usability testing comprised of five main activities which are planning the session, recruiting participants, designing the tasks, running the session and analyzing the insights.The usability testing was conducted by molecular biologist (masters and PhD students) from NM-AIST and using the latest prototype SSAAT a web application hosted at the NM-AIST server. The session was done remotely through Google Meet teleconferencing platform whereby both the moderator and participants were present online during the session. The study conducted remotely to due to the geographical locations between the moderator and participants. The moderator was responsible for starting the session and providing the test cases and tasks instruction. Also, apart from that the moderator was supposed to observe each participant navigation choices, number of tasks completed correctly, number of incomplete tasks, and collect feedbacks and comments while participant was doing usability test.

3.8.1 Usability Testing Methodology

In this study both qualitative and quantitative methods were used to capture user interactivity to the web application. Qualitative data was collected through Likert scale questionnaire whereby quantitative data such as total users who were able to complete all the tasks, total complete tasks, complete task time and others were collected as questionnaires. The testing session was design in way that each individual participant performs all the tasks and summative assessment was done with an intension to examine and evaluate participants insights. The study aimed at capturing the indicating factors to usability such as learnability, efficiency, usefulness and satisfaction through the test session conducted.

3.8.2 Participants and Durations

Total of fifteen (15) participants with age 21 years old and above participated the usability test sessions this is based on previous studies on usability that 5-20 participants is valid for usability testing (Macefield, 2009). Among the participants eight of them were females and seven males. First three participants scheduled on first day were regarded as pilot for the next sessions. The testing sessions were conducted for five day starting from 26th October 2020 to 30th October 2020 with three participants per each day. The duration for each session was 60 minutes as and there was break period for 60 minutes every after one session ends. The session time is based

on previous study on usability that suggest 60-90 minutes is valid for test sessions (Ju & Onse, 2005).

During the testing sessions the moderator provided brief overview about the test session and requested the participants to fill in a pre-test questionnaire so that to collect the general data. Next participants read the task instructions and begins to perform the tasks on the tool using through the web browser. As soon as participants completes all the tasks, the moderator requested the participants to rate the web application (SSAAT) using a Likert scale questionnaire. Lastly, the post-test session follows to find out more information about the overall web application. The following are some of the questions which were asked during the post-test session.

(i) What did the participants like most about the web application?

(ii) What is the challenging part of the web application?

(iii) What should be done to improve?

3.8.3 Tasks

From the table below are the tasks that were designed by the moderator and supplied to the test participants to attempted completion during the usability testing sessions. Each participant was required to attempt the tasks and the moderator was observing, recording the time of completion and other participant's behavior while attempting the task during the session. Table 5 depict the usability testing task with a corresponding allocated time.

Table 5: Task for Usability Testing

Tasks	Estimated Time (min) / baseline
Identify the use the web application.	5 / 3
File upload, view the sequence quality and download the extracted sequence as FASTA file.	5 / 3
Navigate to Chromatogram Viewer, trim the 5'end 60 base and trim 3' end 100 base the download the chromatogram as PDF file	10 /7
Upload a reference sequence and calculate the global/local sequence alignment with the previous uploaded file as a primary sequence.	10 / 7
Generate a report with sequence detail, chromatogram quality score plot and sequence alignment results	5 / 3

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Developed Tool

Studies indicate there are few free open source software tools that deal with analysis of Sanger sequence DNA analysis. Among the few available tools most are command-line based tools, some were desktop stand-alone graphical user interface application while others are web-based applications.

The command-line based tools subject biologists to steep learning curves since they require assistance from computer literate personnel since the nature of interaction of the software tools is based on computer commands (Feizi, 2013; Feizi & Wong, 2012). Tools that had graphical user interface were categorized as desktop stand-alone and web-based application. The tools had a less learning curve since any user can interact with them easily through graphics (Strecker & Memon, 2011).

However, among the tools with graphical user interface there were some limitations such as interoperability issues for desktop stand-alone tools and also lack integrated functionalities across most of the available tools. Studies revealed that some of stand-alone tools only operates in a specifying operating system that may require user to have similar computing environment to install and use the tool. On the other hand, the analysis shows that most of the available tools lacks integrated functionalities since they were built for specific task only and this requires users to more application install to accomplish their works. Table 6 shows summarize finding results for the Sanger sequence analysis existing tools.

4.2 Developed Tool Results

Sanger sequence analysis tool was successfully developed in R programming language, whereby Shiny (Ishimaru *et al*., 2014) web application framework was used to build a graphical user interface in R. SSAAT prototype is hosted on the on the NM-AIST web server and can be accessed through the link http://41.59.85.220/ssaat/. The development of SSAAT on a web application enhances users with limited programming skills to analyze and visualize their data and different users can access the tool concurrently. Users distributed remotely were able access the tool via the Internet network using the web link provided.

Table 6: Finding Results for the Sanger Sequence Analysis Existing Tools

Tool	Programming Language	Type	Availability	Base-calling	Sequence alignment	Chromatogram Viewer	Sequence Quality	Polymorphism Detection	Report generation
SangerSeqR	R	Command-Line	Free, available	Yes	Yes	No	No	No	No
Tracy	C++	Command-Line	Free, available	Yes	Yes	No	No	Yes	No
Phred	Perl	Command-Line	Not free, available	Yes	No	No	Yes	No	No
ASAP	Python	Command-Line	Free, available	No	Yes	No	No	No	No
CHILD	Perl	Web application	Free, Not available	No	No	No	No	Yes	Yes
seqTrace	Python	Desktop	Free, available	Yes	No	Yes	Yes	No	No
Shiftdetector	Perl	Web application	Free, available	No	No	No	No	Yes	No
Indelligent	Perl	Web application	Free, available	No	No	No	No	Yes	No

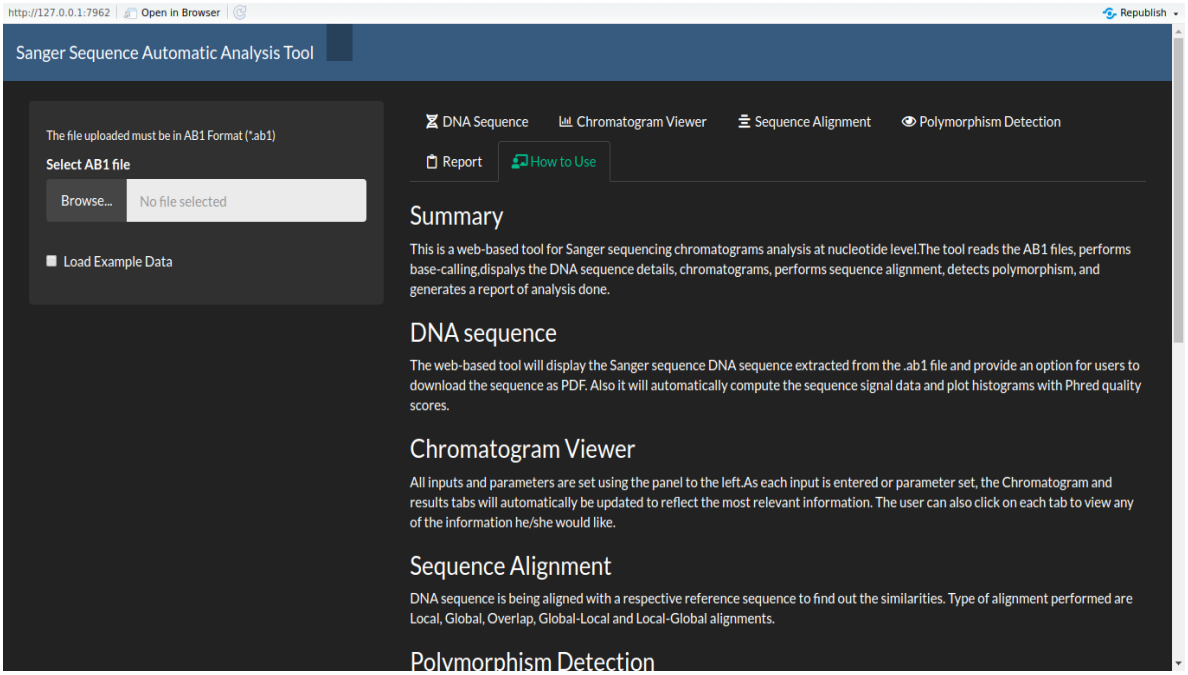


Figure 13: Sanger Sequence Automatic Analysis Tool (SSAAT) Main Page

4.3 Features of the Developed Tool

The SSAAT was developed with six major features one being capabilities to read Applied Biosystems AB1 file format and extract the raw DNA data. Additionally, the tool was capable to perform base-calling, display chromatogram with polymorphism detection, perform sequence alignment, and generate analysis report.

4.3.1 Reading AB1 Files

Sanger sequence analysis tool was able to read AB1 files, the DNA file format are ones processed by Applied Biosystems DNA sequencer machine (Schema, 2009). The AB1 files were uploaded through the graphical upload field in order for the files to be read by the SSAAT AB1 program (Fig.14). The tool was able to access the binary raw data and extract the data. The extracted data was later used for analyses such as displaying the base-call, plotting chromatogram traces, DNA sequence extraction in FASTA file format, sequence alignment and for polymorphism analyses. Moreover, SSAAT was developed with a built-in example of AB1 file for demonstration purposes. Users may learn on how to use the tool by checking in the checkbox to load the example datasets. The SSAAT was able to automatically read the example AB1 file for further analyses tasks.

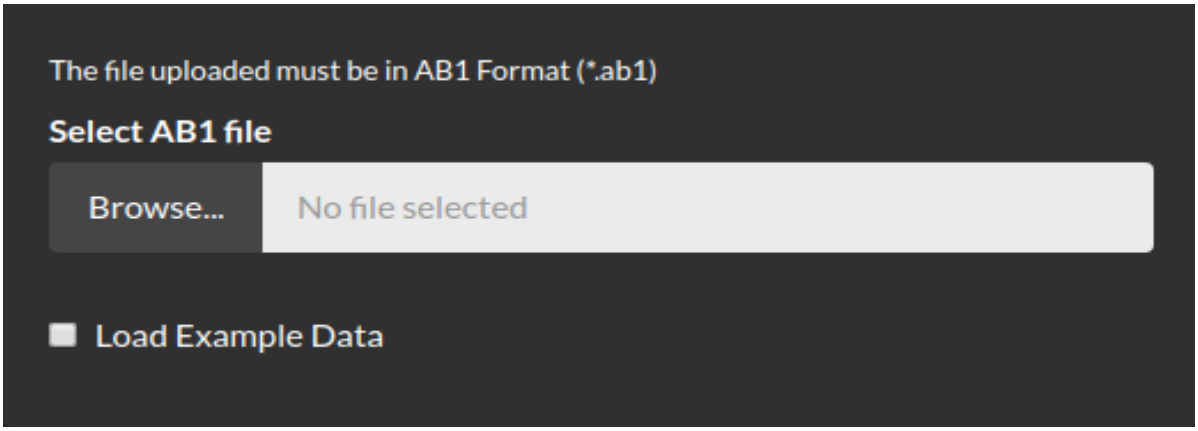


Figure 14: Applied Biosystems File Format File Upload Field Interface

4.3.2 Base Calling

Sanger sequence analysis tool was able perform base-call and extract Sanger DNA sequence and details of the AB1 file. Figure 15 below shows the DNA sequence extracted from the AB1 file with and SSAAT provided an option below for download as FASTA format.

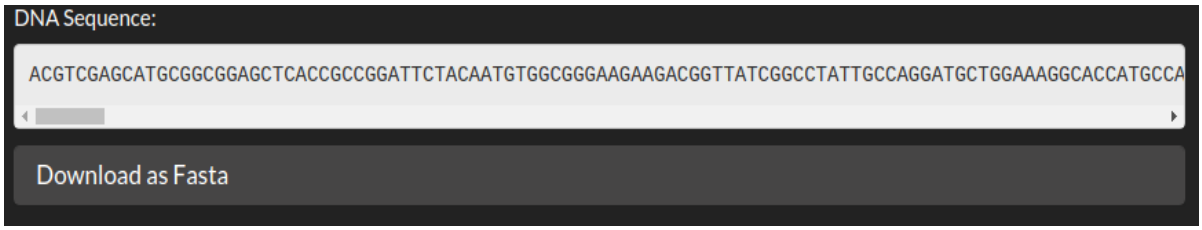


Figure 15: Deoxyribonucleic Acid Sequence Extract by SSAAT

Moreover, SSAAT was able to capture the Phred quality score for AB1 file and plot histogram and cumulative histogram to show the relationship between the nucleotides and Phred quality score. As previously discussed in Chapter Three, Phred quality scores are important to be checked for every processed AB1 files for the purpose of base-call verification (Roberts, 2011). Figure 16 shows how SSAAT was able to capture the Phred quality scores and plot a histogram with a relationship to the samples (nucleotides) bases.

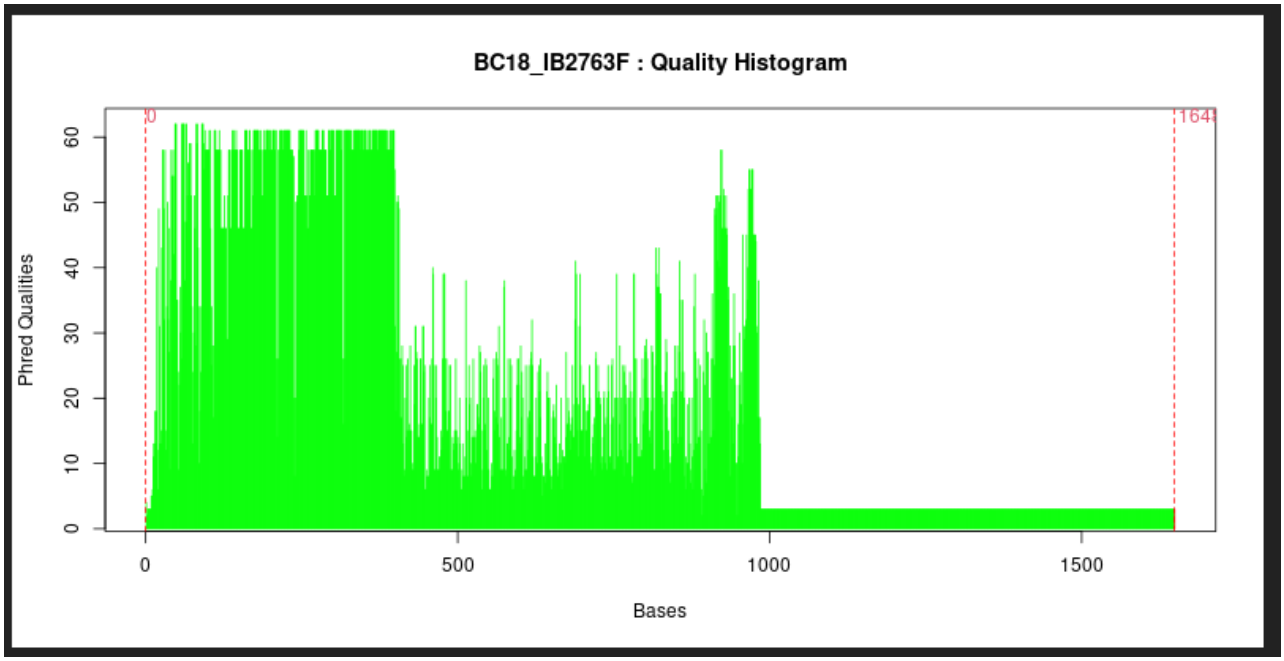


Figure 16: Chromatogram File Quality Score

4.3.3 Chromatogram Viewer

Sanger sequence analysis tool was developed with a chromatogram viewer with the aim of visualizing the chromatogram trace files. The visualization takes as soon as base calling process completes and the nucleotides are assigned to the chromatogram peaks. The user has the ability to change the default settings. Next-Gen option provided by the chromatogram viewer. Below are the chromatogram described in detail.

4.3.4 Chromatogram Width

This setting enables the user to set the number of bases per row on the chromatogram viewer. Sanger Sequence Automatic Analysis Tool provides a slide which default is being set to 100 bases per row, the user is able to slide above to 400 base per row or below to 10 base per row in order to set a desirable number of bases for visualization. Figure 17 and Fig. 18 shows how the can be visualized when the chromatogram width is set to accommodate 100 and 400 base per row respectively. It was found that chromatogram traces tend to widen out and hence clearly visible when the number of bases per is decreased. On the other hand, increasing the number base per row narrowed the traces which resulted to unclear chromatogram for visual investigation.

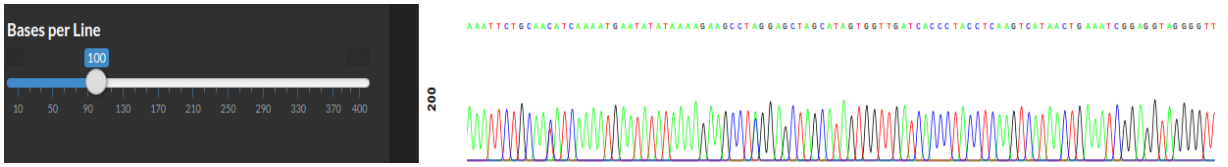


Figure 17: 100 Base Per Line Chromatogram Setting

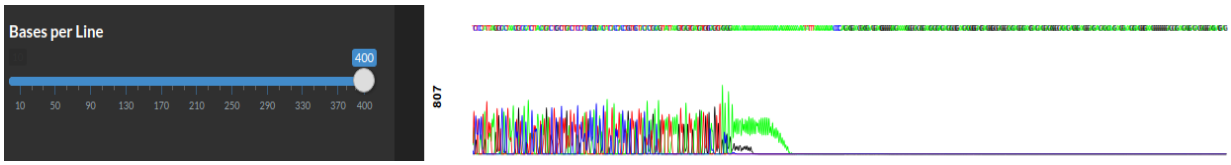


Figure 18: 300 Base per Line Chromatogram Setting

4.3.5 Trimming

Applied Biosystems file format (AB1) trace files usually contain noisy signal output located at the 5 prime region which is beginning and at the 3 prime region which is the end of trace file. SSAAT was developed with trimming setting option to trim the end of the trace files so that to eliminate noisy part of the trace file. Sanger sequence analysis tool trimming came of two modes one being a manual whereby user specify the trimming regions, while the other one was automatically trimming the noisy region. When applied the trimming setting was successfully executed and noisy region of chromatogram were removed.

4.3.6 View Trimmed Region

Sanger sequence analysis tool was able to provide an option setting to choose whether to display the trimmed region of the trace file or not to display. This setting was important for the user to visualize the regions on the chromatogram they are trimming so that they could trim the unwanted part only. Most cases without visualizing the region to be trimmed one could end trimming the region of interest in the chromatogram trace file. Figure 19 shows how SSAAT displays trimmed region of the chromatogram trace file by highlighting the removed bases with crossed red lines in the chromatogram viewer.

Figure 19: Automatic Trimming Option and a Display Removed Bases

4.3.7 Sequence Alignment

Sequence alignment is the process of arranging and comparing DNA sequences to identify regions of similarity that may be a result of functional, structural, or evolutionary relationships between the sequences. Usually, this process involves two for pairwise alignment and more than two DNA sequences for multiple sequence alignment.

Sanger sequence analysis tool was designed to perform pairwise sequence alignment whereby a user is required to upload the Sanger DNA sequences and a reference sequence file in FASTA format. SSAAT provides options for the user choose the type of sequence alignment for the uploaded Sanger sequence data. The following are the alignment method options provided by SSAAT.

4.3.8 Local Alignment

This method of sequence alignment implements the Smith-Waterman algorithm useful for unrelated sequences that are suspected to contain regions of resemblance sequence motifs within the context of larger sequence.

4.3.9 Global Alignment

This method of alignment aligns every residue in every sequence, and it most useful when the sequences in the query set are similar and of roughly equal size. It implements the Needleman–Wunsch algorithm.

4.3.10 Glocal Alignment

Glocal stands for Global-local which is a hybrid type of sequence alignment which combines the use of global and local alignments. Glocal alignment performs searches for the best

probable biased alignment among the sequences.

4.3.11 Polymorphism Detection

DNA polymorphism refers to the difference sequences of among an individual, group or population (Teama, 2018a). At nucleotide level the types of polymorphism include single nucleotide polymorphism (SNPs), insertion/deletion (InDels), variable number of tandem repeats, structural alterations and copy number variations. In this study SSAAT was designed to detect SNPs and InDels types of Sanger sequence DNA polymorphism.

Single nucleotide polymorphism are "high-density natural sequence variations in human genome" formed when errors occur substitution, insertion and deletion (Xu *et al.,* 2009 ;Teama, 2018b). Single nucleotide polymorphism are outstanding sources of variation in human genome and serve as first-class genetic markers. InDels are type of DNA variation in which a particular nucleotide sequence of various lengths ranging from one to several hundred base pairs is inserted or deleted. Mostly, InDels are widely spread across the genome (Boltz *et al*., 2013).

Sanger sequence analysis tool was able to detect SNPs of through the chromatogram viewer and highlight the polymorphic region in the chromatogram. Figure 20 below shows SSAAT detected and highlights the SNPs of the analyzed chromatogram traces files. The observed SNPs from the chromatogram viewer can be verified through sequence alignment at the polymorphism detection section of SSAAT. InDels in SSAAT are being detected through Smith-Waterman algorithm. Sequences to be tested for polymorphism is being aligned with a reference sequence and SSAAT computes the InDels.

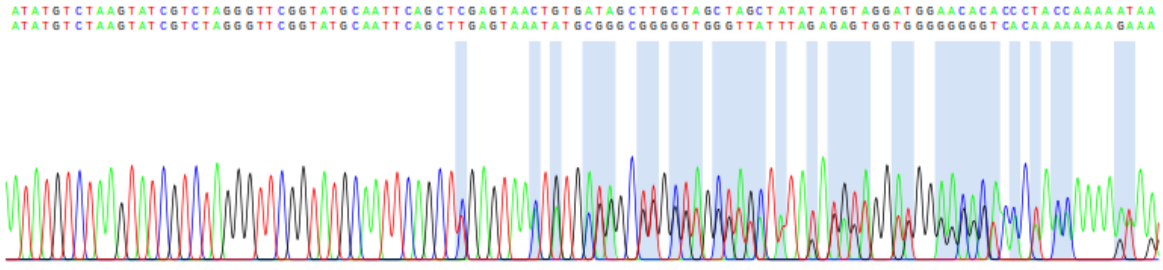


Figure 20: Single Nucleotide Polymorphism (SNP) Highlighted in Chromatogram Viewer

4.3.12 Report Generation

Sanger sequence analysis tool was able to provide report of the analysis done for a particular Sanger DNA sequence file. It was able to provide a report on a sequence alignment information

such as the DNA sequence used, reference DNA sequence, length of sequences, gap and extend penalty used, identity and similarities in percentage, gaps found and the alignment scores. Moreover the DNA sequence extracted with its respective chromatogram qualities scores plots may also be generated by SSAAT as shown in Fig. 21 below. The report generation page provides options for download in three different file formats PDF, Word and HTML.

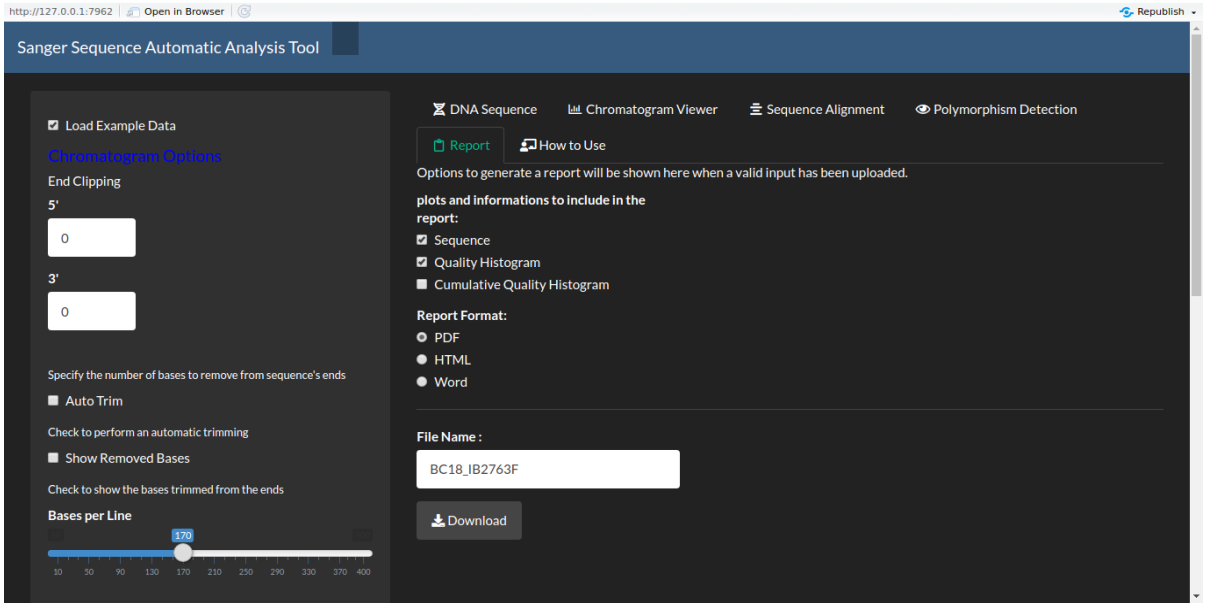


Figure 21: Report Generation

4.4 Validation Results

Sanger sequence analysis tool was developed with user center in mind. It has an easy-to-use web graphical user interface through which users may have easy interactions with the system.

4.4.1 Unit Testing Results

The unit test was conducted during the development stage where different test cases were designed and implemented. The purpose of this test was to validate that each unit of the software performs as designed. The good the unit tests are written the better the results of promptly caught defects introduced due to the change. Also, making code less interdependent during unit testing, makes the unintended impact of changes to any code is less. Table 7 shows the unit test cases conducted.

Table 7: Sanger Sequence Analysis Tool Unit Test Cases

Features	Test Cases	Status
AB1 file upload	Check if AB1 files can be uploaded to the web	OK

	application	
AB1 file read	Check if AB1 file can be read or not	OK
Base-call	Check if base-caller program can assign nucleotides to chromatogram peaks or not	OK
DNA sequence extraction	Check if the DNA extractor can extract the DNA sequence details from base-caller or not	OK
Chromatogram view	Check if chromatogram viewer can display the DNA traces or not	OK
Chromatogram viewer	Check if the chromatogram viewer are working or not	OK
Sequence alignment	Check if the sequence aligner program can perform the DNA sequence alignment or not	OK
Sequence alignment	Check if the sequence aligner are working or not	OK
Polymorphism detection	Check if polymorphism detector is working or not	OK
Report generation	Check if the report generator is working or not	OK
User Manual page	Check if the user manual guide page is displayed or not	OK

From Table 7 above it is indicated that test cases were written to focus on the tests that impact the behavior of the system. The unit test was conducted to validate the components' behaviors using test data that is with different qualities ranging from high to low. Unit test was important in this study so as to reduce cost of fixing defect codes at component level rather in the system level.

4.4.2 Integration Testing Results

Integration testing was conducted with the aim of exposing faults in the interaction between integrated units. The test was intended for verifying the smallest modules validated in the unit testing stage above can work together properly and verify if they are in line with requirements

specified in the software design (Grechanik & Devanla, 2016). All functional units were tested and integrated to verify if they can work together properly. During development base-call module being the heart of the tool was integrated with other modules and tested to verify if they were working properly. This was done so as to evaluate the compliance of each module with specified functional requirements (Grechanik & Devanla, 2016).

4.4.3 System Testing Results

System testing in this study was conducted to check the complete working web application. This test was done after the unit and integration test was conducted so as to validate the systems functional requirements as shown in Table 8. All the functional requirement passed the system testing and hence the developed tool was able to meet the expected requirements.

Table 8: System Testing Results

Function Requirement	Test Cases	Status
Upload DNA sequence data files	Check if AB1 files can be browsed from local machine and uploaded to the web application	Pass
Perform Base-calling	Check if base-caller program can assign nucleotides to chromatogram peaks or not	Pass
DNA sequence extraction	Check if the DNA extractor can extract the DNA sequence details from base-caller or not	Pass
Display chromatogram	Check if chromatogram viewer can display the DNA traces or not	Pass
Chromatogram settings	Check if the chromatogram viewer are working or not	Pass
Convert AB1 file to FASTA format	Check if the tool can convert AB1 file to FASTA format or not	Pass
Extract DNA sequence file details	Check if the system can extract DNA sequence file details or not	Pass
Sequence alignment	Check if the sequence aligner program can perform the DNA sequence alignment or not	Pass
Polymorphism detection	Check if polymorphism detector is working or not	Pass
Report generation	Check if the report generator is working or not	Pass
Operational manual/instruction	Check if the user manual guide page is provided or not	Pass

4.4.3 System Performance Testing Results

The system performance testing was done to measure the remote connectivity between the client-side and the hosting servers. The test was done by auditing the performance factors such as first content paint, speed index, largest content paint, time to interactive, total blocking time and cumulative layout shift (Google, 2015) as presented in Table 9. The performance testing

metric was measured in milliseconds. Google PageSpeed Insight (Google, 2015) testing platform was used to test SSAAT performance and the SSAAT was able to scored 88% in desktop browsing and 64% in mobile device browsing overall performance score. The results from Apache JMeter (Apache, 2014) SSAAT had almost similar performance to those from Google PageSpeed Insight with a score of 86% for desktop browsing and 66% for mobile browsing. Figure 22 shows the SSAAT performance results for a desktop browsing environment.

Table 9: Performance Testing Audits

Web application performance audit	Weight
First content paint	15%
Speed index	15%
Largest content paint	25%
Time to interactive	15%
Total blocking time	25%
Cumulative layout shift	5%

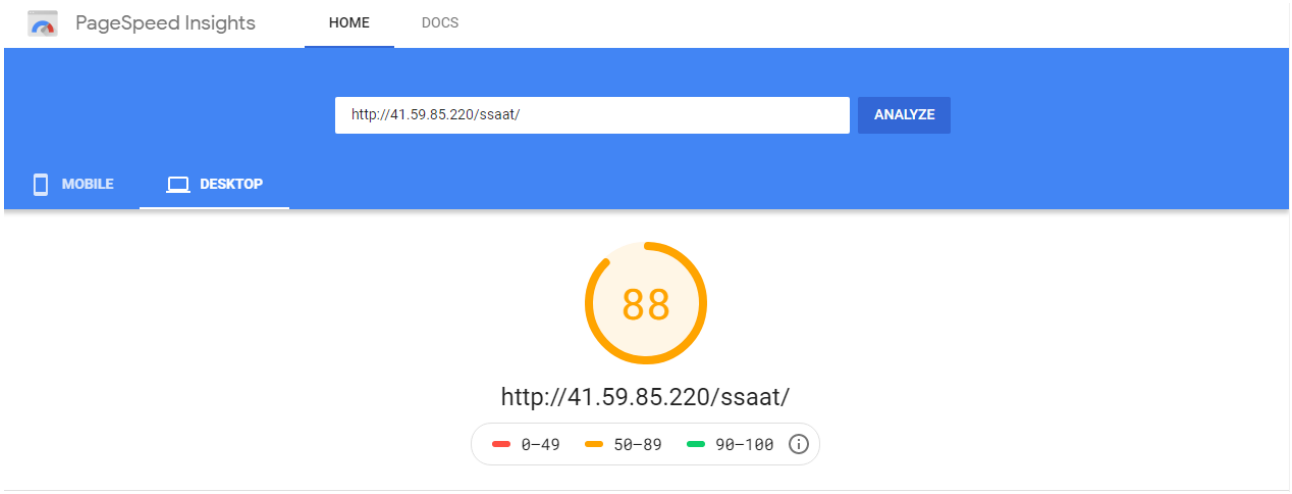


Figure 22: Sanger Sequence Analysis Tool Performance Testing for Desktop Device

4.4.4 Compatibility Testing Results

Compatibility test result shows that SSAAT web application is 85% compatible with computers both desktops and laptops while scores 66% compatibility with mobile devices. The result suggests best user experience in computers more than in mobile devices. This may be due to high screen resolution in loading web pages by computers as compared to mobile devices. Figure 23 shows the page load speed results of SSAAT across five common web browsers

Mozilla Firefox, Google Chrome, Microsoft Edge, Safari and Internet explorer. The test results obtained are for both desktop computer and mobile devices such as phablet and mobile phones.

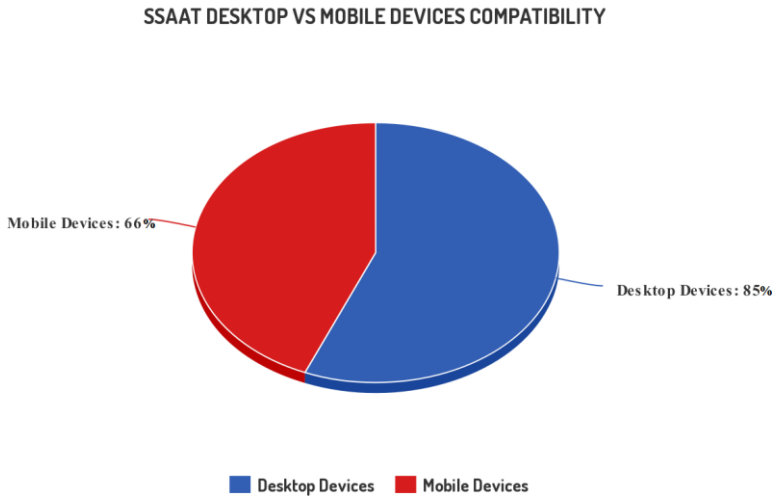


Figure 23: Sanger Sequence Analysis Tool Compatibility Testing Results

4.4.4 Usability Testing Results

(i) Task Completion Rate

All participants successfully completed Task 1 (Identify the use the web application.) marking 100% completion rate. On the other hand, Task 2 (File upload, view the sequence quality and download the extracted sequence as FASTA file.) was completed by 87% of total participants. Among the long duration tasks, Task 3 (Navigate to Chromatogram Viewer, trim the 5'end 60 base and trim 3' end 100 base the download the chromatogram as PDF file) scored 73% completion rate while Task 4 (Upload a reference sequence and calculate the global/local sequence alignment with the previous uploaded file as a primary sequence.) scored 60% completion rate. Moreover, Task 5 (Generate a report with sequence detail, chromatogram quality score plot and sequence alignment results) was successful completed at rate of 73%.

Table 10: Usability Test Tasks Completion Rate

Participants	Task 1	Task 2	Task 3	Task 4	Task 5
Participant 1	✓	✓	-	-	✓
Participant 2	✓	✓	✓	-	✓
Participant 3	✓	✓	✓	✓	-
Participant 4	✓	✓	✓	✓	✓
Participant 5	✓	✓	-	-	✓
Participant 6	✓	✓	✓	-	✓
Participant 7	✓	-	✓	✓	✓
Participant 8	✓	✓	✓	✓	✓
Participant 9	✓	✓	✓	✓	-
Participant 10	✓	✓	✓	✓	✓
Participant 11	✓	✓	✓	✓	✓
Participant 12	✓	✓	✓	-	-
Participant 13	✓	✓	✓	✓	-
Participant 14	✓	✓	-	✓	✓
Participant 15	✓	-	-	-	✓
Success	15	13	11	9	11
Task Accomplishment Success rates	100%	87%	73%	60%	73%

(ii) Mean Time Used on Each Task

The moderator recorded the task execution time for each user participant. The allocated time for each task ranged from 5 to 10 minutes where by simple tasks were allocated less time and lengthy tasks had more times respectively. Task 1 had the shortest time for completion with a mean time of 56 seconds followed by Task 5 and Task 2 with time of 3.1 minutes and 4 minutes respectively. Task 3 and Task 4 obtained the longest time to complete with mean time of 7.3 minutes and 9 minutes respectively. Over all the completion time ranged from 56 sec to 9 minutes and a commonly time recorded was less than 5 minutes.

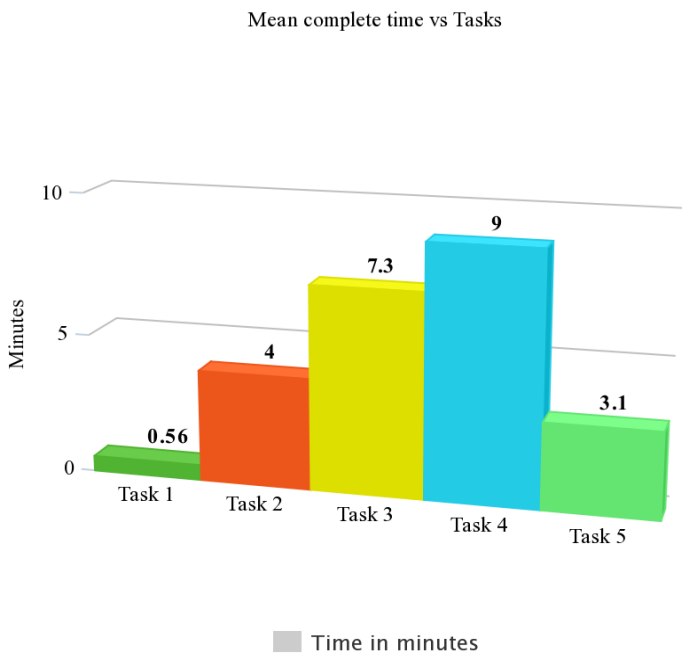


Figure 24: Mean Completion Time Versus Tasks

(iii) System Usability Scale (SUS) Results

During the post-testing session, participants were asked to rate the web application tool to capture general usability measures. Below listed are the measures which were captured from test participants post-test questionnaire.

- If the users would prefer to use the web application tool
- Ease of use
- Learnability
- Assistance from technical personnel
- System functionality integrations
- Recommendation a tool to a colleague

Majority of the participants 86.7% agreed (Agree or Strongly Agree) the web application tool was easy to use. Additionally, most participants 93% agreed they would prefer to work with the web application often. Regardless of the higher percentages of participants agreeing the tool was easy to use, 40% of them agreed that it a technical assistance is requires for them to operate well the tool. productive using this system. More than 66% participant agreed that the integrated functorialities were functioning well while more three quarter of the participant agreed they would recommend the tool to a colleague.

Table 11: System Usability Scale (SUS) Results

Post usability test questionnaire (System Usability Scale)	Respondents					Percentage Agree
	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	
The web application is user friendly	0	0	2	8	5	86.67%
I would like to use web application often	0	0	2	10	4	93%
I think most of the users will be able the fast how to use the web application	0	2	1	8	4	80%
I will need the help of a technical person to be able to use the web application	0	7	2	4	2	40%
I think the web application units/parts are well integrated	0	1	4	6	4	66.67%
I will recommend this web application to my colleagues	0	0	2	9	4	86.67%

4.4 Discussion

The finding of this study shows that most of the free open source DNA analysis software tools particularly those for Sanger sequencing are command-line based which have a steep learning curve for most users. The command-line based tools require advanced computer literate users since they can only be operated through commands with which most of the other users are not well familiar (Feizi & Wong, 2012). The study also revealed that there exists also free open source software tool for Sanger sequence that offered graphical user interface to users but still it would require one to have several applications to perform dry laboratory activities since the tools were limited to specific functionalities only. Interoperability issue was also among challenges possessed by some of the open source tools which were only limited to the specific operating system and hence provided room for usage to users with a similar computing environment. This leads to a limitation that requires users in order to use a particular software tool must migrate first from the current operating environment to the one suitable for the software tool to run. As a result, the migration may lead to loss of time and sometimes even valuable user's data may get lost if back-up of data was not done.

The study furthermore presented the proposed and developed software tool prototype SSAAT as an attempt towards a solving challenge previously faced by open source software tools users. The tool prototype was developed as a web application and it provided both a graphical user interface and at the same time solved the interoperability across the computing environment and as a result more users may be able to access and use the tool. Apart from that, the tool presented consisted of more than integrated DNA analysis functionalities as an attempt towards a one-stop point application.

The Usability validation results of the proposed software tool prototype show a good response from users towards adapting and using the developed tool through the scoring of more than 85% for Systems Usability Scales (SUS) results. The SUS results scored from 80.3% and above, suggest that a product or service is of good quality and may be recommended to other users (Hosio *et al*., 2014). The results also spotted that most of the users liked the user interface of the tool and would like to use it often for themselves SUS score 93%, and this score is a good indication that users were satisfied with the added usability and are willing to move towards the online tool. Moreover, more than half of the users found that it was easy for them to work with the tool without help from technical personnel and also all users were able to

complete all test session's tasks within the provided time. This suggests that the tool had a fair learning curve for most of the users who used it for the first time.

In addition, the tool also achieved good performance of the online connectivity and page loading speed both in computer devices and mobile devices with the assumption that there is a stable connection. The device compatibility results suggest the tool will suitably be accessed through computer devices such as laptops or desktop computers more than mobile devices. This could be due to the small size screens of the mobile devices not able to display properly the tool's graphics.

However, apart from the good usability results obtained from this study, the users suggested a number of possible improvements to the tool such as the incorporation of batch processing capabilities, more options on the chromatogram viewer such as the chromatogram editing, and integrating remote DNA databases. The users suggestions were important towards improving the tool nevertheless, they would require more development effort and time, therefore we plan to work on them in the future versions. Other modifications such as the suggested lighter interface background colours and the creation of a user guide with some visual illustrations were easier and more straightforward to implement.

Furthermore, there were some challenges during the usability testing; some test sessions took longer due to the problem of internet connectivity. The sessions were conducted online through a teleconference application via internet connection since participants were remotely located. Also, some participants appeared late in the test session, which led to rescheduling of other test sessions.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The preceding chapter presented the results and discussion of this study. It describes the findings from the literature about the availability existing open source Sanger sequencing analysis software tools including their features such as command-line based, desktop stand-alone application, and web based. The existing open source tools were also found with limitations such as having steep learning curve, lack of cross platform interoperability, some being open source but not free available and lastly, lack of integrated functionalities.

The author presented results of the proposed and developed software tool and the analyses results of the tool validation. This study made an attempt to develop a free and user-friendly web application for Sanger sequence analyses named Sanger Sequence Automatic Analysis Tool (SSAAT). The development of SSAAT done as an attempt to address some of the challenges possessed by the existing tools that includes empowering users with friendly interaction with their software tools by building graphical user interface (GUI). This would decrease the steep learning curve for using and interacting with the tool by the biologist and this will save time and resources required for analyses.

Sanger sequence analysis tool development also attempted to address the cross-platform interoperability issues through building the software tool as a web application which does not installation rather can be accessed through internet browser. In addition to that, the web application would facilitate sharing of computation resources by more than one user and could be accessed remotely by more than one user at a time.

Additionally, through the integration of several DNA analysis functionalities such base-calling, chromatogram display, polymorphism detection, sequence alignment and reporting SSAAT made an attempt address the lack of functionalities limitation. This an attempt made SSAAT to became a one stop point application (Smith, 2014). This feature enable user to rely on a single application while working and hence increase concentration while saving time and other resources (Smith, 2014).

Moreover, SSAAT was developed with free and open source in mind and hence will not only be available freely to used but also the source codes will publicly available for other scholars

to study and develop other software tools from the base codes. This is an attempt towards eliminating barriers to availability of free and open source software for Sanger sequence DNA analyses.

Usability assessment results suggest that most of users will be able to use the software tool without assistance. This is a good indicator that the tool is easy to use and hence most of the users a likely to often use then tool. Most of the users liked to use SSAAT in their works and would recommend to colleagues. However, usability test results also highlighted that few of the user found SSAAT functionalities not well integrated and some of the would require a technical assistance while working with the tool.

Some limitations of SSAAT one is the lack of batch processing capabilities. The tool only compute analyses for DNA files one at a time. This feature is important especially if the user has more than one Sanger sequence DNA files to work with at time. Currently, SSAAT only support analysis of single file at a time as an initial prototype aiming at address the existing tools limitation first. Other limitation is the capability to store user data in database and security mechanism to protect data sharing and accessing from the database. In future this are among important features that will be incorporated for the next version of SSAAT.

5.2 Recommendations

The Sanger sequencing technique is still a very important method for the molecular biology research studies up to date due to its high accuracy as compared to other methods. It is clear that this DNA sequencing technique will still exist even in the future to come and hence more investment is required to enhance the method. In the software world Sanger sequence analysis software tools lags behind and especially for the free and open source software. There comes a necessity to have an active open source community for developing, documenting and sharing software and methods for Sanger sequence across the globe and especially in Africa where we current need more scientist to handle our local matters.

It is recommended to bioinformaticians to put more effort in the development tools to Sanger sequence tools as it is done for the of Next Generation sequence analysis software. Reusability of the existing software will increase the usage of the free open-source software tools for Sanger sequence analyses.

The future work of this study will focus on improving SSAAT by incorporating batch processing to the tool so that more than one DNA files may analyzed at a time. Moreover, database connectivity is also important so as to be able to store the processed and raw data files, share and manage data at the same time providing security mechanism to protect how to access and share data. These features will increase the efficiency of SSAAT by enabling users to work with analyses of large Sanger sequence data sets within a minimal time with a secured storage capability.

REFERENCES

Abate, A. R., Hung, T., Sperling, R. A., Mary, P., Rotem, A., Agresti, J. J., Weiner, M. A., & Weitz, D. A. (2013). DNA sequence analysis with droplet-based microfluidics. *Lab on a Chip*, *13*(24), 4864–4869. https://doi.org/10.1039/c3lc50905b

Allaire, J. J. (2015). *RStudio: Integrated development environment for R.* https://www.r-project.org/conferences/useR-2011/abstracts/180111-allairejj.pdf

Apache. (2014). *Apache JMeter - Apache JMeter™*. *Apache JMeter*. https://jmeter.apache.org/

Barsch, S., Klein, A., & Verstraeten, P. (2016). *About the authors. In The Imperfect Historian*. https://doi.org/10.3726/978-3-653-03016-7/8

Beeley, C. (2013). *Web Application Development with R using Shiny. In Surveillance and Society*. https://doi.org/10.1017/CBO9781107415324.004

Applied Biosystems Genetic Analysis .(2006). *Data File Format Subject: ABIF File Format Specification and Sample File.* https://projects.nfstc.org/workshops/resources/ articles/ ABIF_File_Format.pdf

Bisandu, D. B. (2019). Design Science Research Methodology in Computer Science and Information Systems. *International Journal of Information Technology*, *11*, 1–7.

Boltz, M., Rau, H., Williams, P., Rau, H., Williams, P., Upton, J., & Remaud, A. (2013). Illness Cognitions and Perceptions. *Encyclopedia of Behavioral Medicine*, *2013*, 1027-1030. https://doi.org/10.1007/ 978-1-4419-1005-9_706

Burzacca, P., & Paternò, F. (2013). Remote usability evaluation of mobile web applications. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8004 LNCS(PART 1).* https://doi.org/ 10.1007/978-3-642-39232-0_27

Chambers, J. (2008). *Software for data analysis: programming with R.* S*pringer Science and Business Media*. file:///C:/Users/user/Downloads/productFlyer_978-0-387-75935-7.pdf

Chang, C. T., Tsai, C. N., Tang, C. Y., Chen, C. H., Lian, J. H., Hu, C. Y., Tsai, C. L., Chao, A., Lai, C. H., Wang, T. H., & Lee, Y. S. (2012). Mixed Sequence Reader: A Program for Analyzing DNA Sequences with Heterozygous Base Calling. *The Scientific World Journal*, *2012*, 1–10. https://doi.org/10.1100/2012/365104

Chmielecki, J., & Meyerson, M. (2014). DNA Sequencing of Cancer: What Have We Learned? *Annual Review of Medicine*, *65*(1), 63–79. https://doi.org/10.1146/annurev-med-060712-200152

Clevenger, J., Chavarro, C., Pearl, S. A., Ozias-Akins, P., & Jackson, S. A. (2015). *Single Nucleotide Polymorphism Identification in Polyploids: A Review, Example, and Recommendations*. https://doi.org/10.1016/j.molp.2015.02.002

Curcio, K., Navarro, T., Malucelli, A., & Reinehr, S. (2018). Requirements engineering: A systematic mapping study in agile software development. *Journal of Systems and Software*, *139*, 32–50. https://doi.org/10.1016/j.jss.2018.01.036

Davis, D., & Jiang, S. (2016). Usability testing of existing type 2 diabetes mellitus websites. *International Journal of Medical Informatics*, *92*, 62–72. https://doi.org/10.1016/j.ijmedinf.2016.04.012

Delseny, M., Han, B., & Hsing, Y. I. (2010). High throughput DNA sequencing: The new sequencing revolution. *Plant Science*, *179*(5), 407-422 https://doi.org/10.1016/j.plantsci.2010.07.019

Dmitriev, D. A., & Rakitov, R. A. (2008). Decoding of superimposed traces produced by direct sequencing of heterozygous indels. *PLoS Computational Biology*, *4*(7), 1-10 https://doi.org/10. 1371/journal.pcbi.1000113

Dorst, K. (2011). The core of "design thinking" and its application. *Design Studies*, *32*(6), 521-532. https://doi.org/10.1016/j.destud.2011.07.006

Ebert, C., Abrahamsson, P., & Oza, N. (2012). Lean software development. *IEEE Computer Architecture Letters*, *29*(05), 22-25. https://doi.org/10.1109/MS.2012.116

Ewens, W. J. (2013*). Genetic Variation. In Brenner's Encyclopedia of Genetics.* https://doi.org/10.1016/B978-0-12-374984-0.00631-8

Ewing, B., & Green, P. (1998). Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Research*, *8*(3), 186–194. https://doi.org/10.1101/gr.8.3.186

Feizi, A., & Wong, C. Y. (2012). *Usability of user interface styles for learning a graphical software application. 2012 International Conference on Computer and Information Science, ICCIS 2012 - A Conference of World Engineering, Science and Technology Congress, ESTCON 2012 - Conference ProceediNext-Gen*. https://doi.org/10.1109/ICCISci.2012.6297188

Fernandez, A., Insfran, E., & Abrahão, S. (2011). Usability evaluation methods for the web: A systematic mapping study. *Information and Software Technology*, *53*(8), 789–817. https://doi.org/10.1016/j.infsof.2011.02.007

Gabarro, S. A. (2015). *Introduction to HTML. In Web Application Design and Implementation*. https://doi.org/10.1109/9780470083963.ch3

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., & Zhang, J. (2004). Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, *5*(10), 1-16. https://doi.org/10.1186/gb-2004-5-10-r80

Goldberg, S. M. D., Johnson, J., Busam, D., Feldblyum, T., Ferriera, S., Friedman, R., Halpern, A., Khouri, H., Kravitz, S. A., Lauro, F. M., Li, K., Rogers, Y. H., Strausberg, R., Sutton, G., Tallon, L., Thomas, T., Venter, E., Frazier, M., & Venter, J. C. (2006). A Sanger/pyrosequencing hybrid approach for the generation of high-quality draft assemblies of marine microbial genomes. *Proceedings of the National Academy of Sciences, 103*(30), 11240-11245. https://doi.org/10. 1073/pnas.0604351103

Gonzaga-Jauregui, C., Lupski, J. R., & Gibbs, R. A. (2012). Human Genome Sequencing in Health and Disease. *Annual Review of Medicine*, *63*, 35-61 https://doi.org/10.1146/annurev-med-051010-162644

Google. (2015). *PageSpeed Insights.* https://developers.google.com/speed/pagespeed/insights_extensions

Grechanik, M., & Devanla, G. (2016). Mutation Integration Testing. *Proceedings - 2016 IEEE International Conference on Software Quality, Reliability and Security, QRS 2016*, 353–364. https://doi.org/10.1109/QRS.2016.47

Greene, E. C. (2016). DNA sequence alignment during homologous recombination. *Journal of Biological Chemistry*, *291*(22), 11572-11580. https://doi.org/10.1074/jbc.R116.724807

Hill, A. J. T., Demarest, B., & Hill, M. J. (2014). *Package 'sangerseqR. http://citeseerx .ist.psu.edu/viewdoc/download?doi=10.1.1.454.2534&rep=rep1&type=pdf*

Hill, J. T. (2015). *Walkthrough for using the sangerseqR package*. https://bioconductor. riken.jp/packages/3.2/bioc/vignettes/sangerseqR/inst/doc/sangerseq_walkthrough.pdf

Hillgren, P. A., Seravalli, A., & Emilson, A. (2011). Prototyping and infrastructuring in design for social innovation. *CoDesign*, *7*(3–4), 169–183. https://doi.org/10.1080/15710882.2011.630474

Hingorani, M. M. (2013). *Template. In Brenner's Encyclopedia of Genetics*. https://doi.org/10.1016/B978-0-12-374984-0.01526-6

Huber, W., Carey, V. J., Gentleman, R., Anders, S., Carlson, M., Carvalho, B. S., Bravo, H.C., Davis, S., Gatto, L., Girke, T., & Gottardo, R. (2015). Orchestrating high-throughput genomic analysis with Bioconductor. *Nature Methods*, *12*(2), 115–121. https://doi.org/10.1038/nmeth.3252

Weppner, J., Poxrucker, A., Lukowicz, P., Ishimaru, S., Kunze, K., & Kise, K. (2014). *Shiny: An activity logging platform for Google Glass. In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. https://doi.org/10.1145/2638728.2638798

Jackson, M., Crouch, S., & Baxter, R. (2011). Software evaluation: criteria-based assessment. *Software Sustainability Institute, 2011*, 1–13. http://software.ac.uk/sites/default/files/SSI-SoftwareEvaluationCriteria.pdf

Jorde, L. B., & Wooding, S. P. (2004). Genetic variation, classification and "race." *Nature Genetics*. https://doi.org/10.1038/ng1435

Lee, N. J., & Bakken, S. (2008). Usability Testing of a Prototype Personal Digital Assistant (PDA)-based Decision Support System for the Management of Obesity. *Perspectives in Nursing Science, 2018, 17-41.*

Kant, T. (2010). Open source bioinformatics workbench options for life science researchers. *New York Science Journal*, *3*(10), 82-87.

Kircher, M., & Kelso, J. (2010). H*igh-throughput DNA sequencing - Concepts and limitations. In BioEssays*. https://doi.org/10.1002/bies.200900181

Koziokas, P. T., Tselikas, N. D., & Tselikis, G. S. (2017). Usability Testing of Mobile Applications. *Proceedings of the 21st Pan-Hellenic Conference on Informatics - PCI 2017*, 1–2. https://doi.org/10.1145/3139367.3139410

Kuhner, M. K., Beerli, P., Yamato, J., & Felsenstein, J. (2000). Usefulness of single nucleotide polymorphism data for estimating population parameters. *Genetics*, *156*(1), 439-447

Kumar, A., & Kumar, A. (2019). *Web Application Frameworks. In Web Technology*. https://doi.org/10.1201/9781351029902-10

Lanka, R., Koti, S., Sunkara, P. S. S., & Undamatla, J. (2014). DNA sequencing analysis software for Sanger data-sets: Comparisons of basic features useful for mutational studies. *Current Trends in Biotechnology and Pharmacy*, *8*(1), 11–17.

Leaché, A. D., & Oaks, J. R. (2017). *The Utility of Single Nucleotide Polymorphism (SNP) Data in Phylogenetics. Annual Review of Ecology, Evolution, and Systematics*. https://doi.org/10.1146/annurev-ecolsys-110316-022645

Leipzig, J. (2017). A review of bioinformatic pipeline frameworks. *Briefings in bioinformatics*, *18*(3), 530-536. https://doi.org/10.1093/bib/bbw020

Luan, P. T., Ryder, O. A., Davis, H., Zhang, Y. P., & Yu, L. (2013). Incorporating indels as phylogenetic characters: Impact for interfamilial relationships within Arctoidea (Mammalia: Carnivora). *Molecular Phylogenetics and Evolution*, *66*(3), 748-756 https://doi.org/10. 1016/j.ympev.2012.10.023

Macefield, R. (2009). How to specify the participant group size for usability studies: A practitioner's guide. *Journal of Usability Studies*, *5*(1), 34–45.

Machado, M., Magalhães, W. C. S., Sene, A., Araújo, B., Faria-Campos, A. C., Chanock, S. J., Scott, L., Oliveira, G., Tarazona-Santos, E. & Rodrigues, M. R. (2011). Phred-Phrap package to analyses tools: A pipeline to facilitate population genetics re-sequencing studies. *Investigative Genetics*, *2*(1), 1-7. https://doi.org/10.1186/2041-2223-2-3

Martin, R., & Euchner, J. (2012). *Design thinking. Research Technology Management*. https://doi.org/10.5437/08956308X5503003

Martins, R. D. S., Campos, J., M., Dos-Santos, M., A., Marques Zembrzuski, V., da Fonseca, A. C. P., Abreu, G. D. M., Cabello, P. H., & De-Cabello, G. M. K. (2019). Identification of a novel large deletion and other copy number variations in the CFTR gene in patients with Cystic Fibrosis from a multiethnic population. *Molecular Genetics and Genomic Medicine*, *7*(7), 1-7. https://doi.org/10.1002/mgg3.645

Maxam, A. M., & Gilbert, W. (1977). A new method for sequencing DNA. *ProceediNext-Gen of the National Academy of Sciences of the United States of America*, *74*(2), 560–564. https://doi.org/10.1073/pnas.74.2.560

Nagaraj, A., Gattu, H., Shetty, P. K., & Professor, A. (2014). Research Study on Importance of Usability Testing/ User Experience (UX) Testing. *International Journal of Computer Science and Mobile Computing*, *310*(10), 78–85.

Nguyen, N. T., Trawiński, B., Fujita, H., & Hong, T. P. (2016). *Responsive Web Design: Testing Usability of Mobile Web Applications. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-662-49381-6

Nidhra, S. (2012). Black Box and White Box Testing Techniques - A Literature Review. *International Journal of Embedded Systems and Applications*, *2*(2), 29–50. https://doi.org/10.5121/ijesa.2012.2204

Oluwatosin, H. S. (2014). Client-Server Model. *IOSR Journal of Computer Engineering*, *16*(1), 2278-8727. https://doi.org/10.9790/0661-16195771

Pagès, H., Aboyoun, P., Gentleman, R., & DebRoy, S. (2017). *BiostriNext-Gen: Efficient manipulation of biological striNext-Gen*. https://bioconductor.org/packages/release /bioc/html/Biostrings.html

Pareek, C. S., Smoczynski, R., & Tretyn, A. (2011). Sequencing technologies and genome sequencing. *Journal of Applied Genetics*, *52*(4), 413-435 https://doi.org/10.1007/ s13353-011-0057-x

Pekin, D., Skhiri, Y., Baret, J. C., Le Corre, D., Mazutis, L., Ben Salem, C., Millot, F., El Harrak, A., Hutchison, J. B., Larson, J. W., Link, D. R., & Taly, V. (2011). Quantitative and sensitive detection of rare mutations using droplet-based microfluidics. *Lab on a Chip*, *11*(13), 2156–2166. https://doi.org/10.1039/c1lc20128j

Pereira, F., Carneiro, J., Amorim, A., & Pereira, F. (2008). Identification of species with DNA-based technology: Current progress and challenges. *Recent Patents on DNA and Gene Sequences*, *2*(3), 187-200. https://doi.org/10.2174/187221508786241738

Pernstål, J., Feldt, R., & Gorschek, T. (2013). The lean gap: A review of lean approaches to large-scale software systems development. *Journal of Systems and Software*, *86*(11), 2797-2821. https://doi.org/10.1016/j.jss.2013.06.035

Poppendieck, M., & Cusumano, M. A. (2012). Lean software development: A tutorial. *IEEE Software*, *29*(5), 26-32. https://doi.org/10.1109/MS.2012.107

Quiñones, D., Rusu, C., & Rusu, V. (2018). A Methodology to Develop Usability/User eXperience Heuristics. *Computer Standards and Interfaces*, *59*, 109-129 https://doi.org/10.1016/ j.csi.2018.03.002

R Development Core Team, R. (2011). R: A Language and Environment for Statistical Computing. *In R Foundation for Statistical Computing Vienna, Austria*. https://doi.org/10.1007/978-3-540-74686-7

Lanka, R., Koti, S., Phani, S. S., & Undamatla, J. (2014). DNA Sequencing Analysis Software for Sanger Data-Sets: Comparisons of Basic features Useful for Mutational Studies. *Current Trends in Biotechnology and Pharmacy*, *8*(1), 11-17.

Rausch, T. (2018). *Workshop: Adopt open standards and interoperable technologies to securely integrate your medical device to clinical environments* https://www.pharma-iq.com/events-sdmdconference/speakers/tracy-rausch

Reese, G. (2000). *Database Programming with JDBC and Java. Database Programming with JDBC and Java.* http://dx.doi.org/10.1016/j.jff. 2011.01.003

Roach, K. (2010). *The Role of Innocence Commissions: Error Discovery, Systemic Reform or Both. Chicago-Kent Law Review*. https://heinonline.org/HOL/Page? handle=hein.journals/chknt85&id=97&div=&collection=

Roberts, M. (2011). *Phred quality score. In Genome*. https://doi.org/10.1101/gr.8.3.175.

Samanta, J., Bhaumik, J., Barman, S., & Maity, R. K. (2017). Binary error correcting code for DNA databank. *Lecture Notes in Electrical Engineering*, *470*, 1–12. https://doi.org/10.1007/978-981-10-8585-7_1

Sanger, F., Nicklen, S., & Coulson, A. R. (1977). DNA sequencing with chain-terminating inhibitors. *ProceediNext-Gen of the National Academy of Sciences of the United States of America*, *74*(12), 5463–5467. https://doi.org/10.1073/PNAS.74.12.5463

Scacchi, W., & Jensen, C. (2012). *Open Source Software Development. In Leadership in Science and Technology: A Reference Handbook*. https://doi.org/10.4135/9781412994231.n88

Schema, S. F. (2009). *Applied Biosystems Genetic Analysis Data File Format SUBJECT: ABIF File Format Specification and Sample File Schema*. *Retrieved from* http://www6.appliedbiosystems.com/support/software_community/ABIF_File_Format.pdf

Seemann, T. (2013). Ten recommendations for creating usable bioinformatics command line software. *GigaScience*, *2*(1), 2047-217X. https://doi.org/10.1186/2047-217X-2-15

Seroussi, E., Ron, M., & Kedra, D. (2002). ShiftDetector: Detection of shift mutations. *Bioinformatics*, *18*(8), 1137–1138. https://doi.org/10.1093/bioinformatics/18.8.1137

Shendure, J., Balasubramanian, S., Church, G. M., Gilbert, W., Rogers, J., Schloss, J. A., & Waterston, R. H. (2017). DNA sequencing at 40: Past, present and future. *Nature*, *550*(7676), 345-353. https://doi.org/10.1038/nature24286

Sillitti, A., & Succi, G. (2005). *Requirements engineering for agile methods. In Engineering and Managing Software Requirements. Springer Berlin Heidelberg*. https://doi.org/10.1007/3-540-28244-0_14

Singh, A., & Bhatia, P. (2016). Automated sanger analysis pipeline (ASAP): A tool for rapidly analyzing sanger sequencing data with minimum user interference. *Journal of Biomolecular Techniques*, *27*(4), 129–131. https://doi.org/10.7171/jbt.16-2704-005

Smith, D. R. (2014). Buying in to bioinformatics: An introduction to commercial sequence analysis software. *Briefings in Bioinformatics*, *16*(4), 700–709. https://doi.org/10.1093/bib/bbu030

Straiton, J., Free, T., Sawyer, A., & Martin, J. (2019). From Sanger sequencing to genome databases and beyond. *BioTechniques*, *66*(2), 60–63. https://doi.org/10.2144/btn-2019-0011

Strecker, J., & Memon, A. M. (2011). *Testing Graphical User Interfaces. In Encyclopedia of Information Science and Technology.* https://doi.org/10.4018/978-1-60566-026-4.ch596

Stucky, B. J. (2012). SeqTrace: a graphical tool for rapidly processing DNA sequencing chromatograms. *Journal of Biomolecular Techniques, 23*(3), 90–93. https://doi.org/10.7171/jbt.12-2303-004

Taylor, J. A., & Kieser, J. A. (2016). *Forensic odontology: Principles and practice. In Forensic Odontology: Principles and Practice. Wiley Blackwell*. https://doi.org/10.1002/9781118864418

Teama, S. (2018a). DNA Polymorphisms: DNA-Based Molecular Markers and Their Application in Medicine. *Genetic Diversity and Disease Susceptibility, 2018*, 5-40. https://doi.org/10.5772/intechopen.79517

Van-Oorschot, R. A. H., Ballantyne, K. N., & Mitchell, R. J. (2010). *Forensic trace DNA: A review. In Investigative Genetics (Vol. 1, Issue 1).* https://doi.org/10.1186/2041-2223-1-14

Vuksanovic, I. P., & Sudarevic, B. (2011). Use of web application frameworks in the development of small applications. *MIPRO 2011 - 34th International Convention on Information and Communication Technology, Electronics and Microelectronics - ProceediNext-Gen*. https://ieeexplore.ieee.org/abstract/document/5967100

Watson, J. D., & Crick, F. H. (1953). Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid. *Nature*, *171*(4356), 737-738. https://www.ias.ac.in/article/fulltext/reso/009/11/0096-0098

Wilson, C. (2013). *User interface inspection methods: A user-centered design method*. Newnes. https://www.amazon.com/User-Interface-Inspection-Methods-User-Centered/dp/01 2410391X

Xu, J. Y., Xu, G. B., & Chen, S. L. (2009). A new method for SNP discovery. *BioTechniques, 46*(3), 201–208. https://doi.org/10.2144/000113075

Zhang, F., Gu, W., Hurles, M. E., & Lupski, J. R. (2009). *Copy Number Variation in Human Health, Disease, and Evolution. Annual Review of Genomics and Human Genetics.* https://doi.org/10.1146/annurev.genom.9.081307.164217

Zhidkov, I., Cohen, R., Geifman, N., Mishmar, D., & Rubin, E. (2011). CHILD: A new tool for detecting low-abundance insertions and deletions in standard sequence traces. *Nucleic Acids Research, 39*(7), e47-e47. https://doi.org/10.1093/nar/gkq1354.

APPENDICES

Appendix 1: Sanger Sequence Automatic Analysis Tool Server Source Code

```
# server.R
library("shiny")
library("Biostrings")
library("sangerseqR")
library("rmarkdown")
library("knitr")
source("ssaa.R")


options(shiny.maxRequestSize = 50*1024^2)
sampleData<-"SampleData/BC18_IB2763F.ab1"

shinyServer(function(input,output,session) {

   sampleName1 <- reactive({
    if(input$example) {
      return(sub("([^.]+)\\.[[:alnum:]]+$", "\\1",basename(sampleData)))
    } else if(!is.null(input$files)) {
      return(sub("([^.]+)\\.[[:alnum:]]+$", "\\1",unlist(input$files$name)))
    } else return(NULL)
  })

 sanger <- reactive({
    if(input$example) {
      obj <- read.abif(sampleData)
      return(obj)
    }
    else if(!is.null(input$files)) {
      obj <- read.abif(input$files$datapath)
      return(obj)
    }
    else
      return(NULL)
  })


  inputdata <- reactive({

    closeAlert(session, alertId="alert")
    if(!is.null(sanger())){
      if (is.null(sanger()@data$DATA.9) ||
          is.null(sanger()@data$DATA.10) ||
          is.null(sanger()@data$DATA.11) ||
          is.null(sanger()@data$DATA.12) ||
          is.null(sanger()@data$PLOC.2) ||
          is.null(sanger()@data$PBAS.2) ||
          is.null(sanger()@data$PCON.2)){
        createAlert(session, anchorId = "invalid_file",
                    alertId="alert",
                    content = "The file uploaded is corrupted or doesn't
contain some important informations.",
                    style = "danger",
                    append=FALSE,
                    block = FALSE,
                    dismiss = FALSE
        )
```

```
        return(NULL)
      }
      else

        return(baseCall_fun(sanger(), sampleName1()))
    }
    else
      return(NULL)
  })

  # Trimming
 seqTrm <- reactive({
    closeAlert(session, alertId="alert2")
    closeAlert(session, alertId="alertthreePrime")
    if (input$autoTrim)
      trimmer <- calcul.trimmer(inputdata()$Q)
    else

      if (is.na(input$trim5) | is.na(input$trim3)){
        createAlert(session, anchorId = "na trim",
                    alertId="alertthreePrime",
                    content = "Can't be NA! Specify valid numbers (default :
0). Or check Auto Trim for an atomatic trimming.",
                    style = "warning",
                    append=FALSE,
                    #block = FALSE,
                    dismiss = FALSE
        )

        trimmer <- list(fivePrime=0, threePrime=0)
      }

    else if( length(inputdata()$seq) <= (input$trim5+input$trim3) ){
      createAlert(session, anchorId = "long_trim",
                  alertId="alert2",
                  content = paste0("Number of bases to trim is greater than
the length of the sequence which equals to ",
                                   length(inputdata()$seq), ". (default :
0)"),
                  style = "warning",
                  append=FALSE,
                  dismiss = FALSE
      )

      trimmer <- list(fivePrime=0, threePrime=0)
    }
    else

      trimmer <- list(fivePrime=input$trim5, threePrime=input$trim3)

    return(trimmer)
  })

  chrmW <- reactive({
    closeAlert(session, alertId="alert4")
    if (is.na(input$x)){
```

```
      createAlert(session, anchorId = "na_bprowth",
                  alertId="alert4",
                  content = "Can't be NA! Enter a valid number (default :
100).",
                  style = "warning",
                  append=FALSE,
                  dismiss = FALSE
      )
      bprow <- 100
    }
    else
      bprow <- input$x
    return(bprow)
  })


  trimmedSequence <- reactive({

    if (sum(seqTrm()$fivePrime ,seqTrm()$threePrime ) >
length(inputdata()$seq))
      return ("")
    else
      return
(inputdata()$seq[(baseTrim()$fivePrime+1):(length(inputdata()$seq)-
baseTrim()$threePrime)])

  })


  chrom_hght <- reactive({
    if (!is.null(input$files) | input$example) {
      chromHeight(inputdata(),seqTrm()$fivePrime,seqTrm()$threePrime,
width=chrmW(), showtrim=input$viewRmBs)
    }
  })


  rfSequence <- reactive({

    if(input$example){
      refseq <- cleanrefseq(input$rfsample)
      return (list(name = "Reference",
                   width = nchar(refseq),
                   seq = refseq))
    }

    if (!is.null(input$refFile)){
      refseq <- readDNAStringSet(input$refFile$datapath)
      return (list(name = refseq@ranges@NAMES,
                   width = refseq@ranges@width,
                   seq = cleanrefseq(toString(refseq))))
    }

    if(input$refseq != ""){
      refseq <- cleanrefseq(input$refseq)
      return (list(name = "Reference",
                   width = nchar(refseq),
```

```
                    seq = refseq))
    }

    else
      return(NULL)
  })


  alignementPw <- reactive({
    if(!is.null(trimmedSequence()) & !is.null(refSequence())){
      mainSeq<-ifelse(input$reverseSeq, char2string(rev(trimmedSequence())),
char2string(trimmedSequence()))
      return(pairwiseAligner(mainSeq,refSequence()$seq,input$align_type))
    }
  })


  output$exportFasta <- renderUI({
    bsCollapse(multiple = FALSE, id = "collapse2",
               bsCollapsePanel("Download as Fasta",
                               textInput("seq_header", label = "Header:",
value = sampleName1()),
                               numericInput("seq_bpl", label = " Number of
Bases per Line:", value = "60", min = 10, max = 100, step = 10),
                               textInput("seq fname", label = "File Name:",
value = sampleName1()),
                               downloadButton('dlfasta', 'Download'))
    )
  })

  output$exportPdf <- renderUI({
    bsCollapse(multiple = FALSE, id = "collapse3",
               bsCollapsePanel("Download as PDF",
                               textInput("chrom_fname", label = "File Name:",
value = sampleName1()),
                               downloadButton('dlchrom', 'Download'))
    )
  })


  output$cbReportSeq<- renderUI({
    checkboxGroupInput("variable1", "plots and informations to include in the
report:",
                       c("Sequence" = "seq",
                         "Quality Histogram" = "qHist",
                         "Cumulative Quality Histogram" = "cumHist"))
  })



  output$ReportFileName<- renderUI({
    textInput("report fname", label = "File Name :", value = sampleName1())
  })


  output$seqDetails<- renderUI({
```

```
    selectInput('in3', '', dumbExtractor(detailSeq(sanger())), multiple=TRUE,
selectize=FALSE,width='100%')
  })


  output$chrom_hght <- reactive(chrom_hght())


  output$flValid<- reactive({return(!is.null(inputdata()))})
  #output options
  outputOptions(output, "flValid", suspendWhenHidden = FALSE)


  output$seqShow <- renderPrint({
    if (!is.null(trimmedSequence()))
      cat(char2string(trimmedSequence()))
  })


  output$AHeader <- renderPrint({
    if (!is.null(alignementPw()))
      cat(alignementPw()$header, sep='\n')
  })

  output$AlignView <- renderPrint({
    if (!is.null(alignementPw()))
      cat(alignementPw()$align, sep='\n')
  })

  output$APack <- renderPrint({
    if (!is.null(alignementPw()))
      cat(alignementPw()$pack)
  })


  output$qHist <- renderPlot({                          #changes here
    if(!is.null(inputdata()))
      plotQHist.fun(inputdata()$Q,
                    paste(sampleName1(),": Quality Histogram",sep=" "),
                    "Phred Qualities",
                   seqTrm()$fivePrime,
                    length(inputdata()$seq)-baseTrim()$threePrime)
  })

  output$cumHist <- renderPlot({
    if(!is.null(inputdata()))
      plotQHist.fun(cum(inputdata()$Q),
                    paste(sampleName1(),": Cumulative Quality Histogram",sep="
"),
                    "Cumulative Phred Qualities",
                   seqTrm()$fivePrime,
                    length(inputdata()$seq)-baseTrim()$threePrime)
  })


  output$chrom.viewer <- renderPlot(
```

```
    chromatogram.viewer(inputdata(),seqTrm()$fivePrime,seqTrm()$threePrime,
                        showOrigin = FALSE,
                        width=chrmW(), height=1, cex.mtext=1, cex.base=1,
ylim=3,
                        filename=NULL, showtrim=input$viewRmBs, showhets=TRUE),
height=reactive(h())
  )


  output$dlfasta<- downloadHandler(
    filename = function() {
      paste0(input$seq fname, '.fasta')
    },
    content = function(file) {
      writeFasta(trimmedSequence(), input$seq header, file, "w", nbchar =
input$seq_bpl)
    }
  )

  # chromatogram download handler
  output$dlchrom<- downloadHandler(
    filename = function() {
      paste0(input$chrom_fname, '.pdf')
    },
    content = function(file) {
      chromatogram.plot(inputdata(),seqTrm()$fivePrime,seqTrm()$threePrime,
                        showOrigin = FALSE,
                        width=chrmW(), height=1, filename=file,
                        showtrim=input$viewRmBs, showhets=TRUE)
    }
  )

  output$ReportFile <- downloadHandler(
    filename = function() {
      paste(input$report fname, sep = '.', switch(
        input$reportFormat, PDF = 'pdf', HTML = 'html', Word = 'docx'
      ))
    },
    content = function(file) {
      src <- normalizePath('report.Rmd')

           owd <- setwd(tempdir())
      on.exit(setwd(owd))
      file.copy(src, 'report.Rmd')

      out <- render('report.Rmd', switch(
        input$reportFormat,
        PDF = pdf_document(), HTML = html_document(), Word = word_document()
      ))
      file.rename(out, file)
    }
  )

   observe({

    if(!is.null(inputdata()))
```

```
    else
      updateTabsetPanel(session, "maintabset", selected = "How to Use")
  })


  observe({

    if(is.null(inputdata())){

      updateNumericInput(session,"trim5",  value = "0")

      updateNumericInput(session,"trim3",  value = "0")

      updateCheckboxInput(session,'autoTrim', 'Auto Trim', FALSE)


      updateCheckboxInput(session,'viewRmBs', 'Show Removed Bases',FALSE)

      updateCheckboxInput(session,'viewOrgBs', 'Show Original Bases', FALSE)

      updateNumericInput(session,"x",  value = "100")

      updateRadioButtons(session,"align_type", selected = "local")

    }
  })


})

# End of server.R
```

Appendix 2: Sanger Sequence Automatic Analysis Tool User Interface Source Code

```
# ui.R

library("shiny")
library("shinythemes")
library("Biostrings")
library("rmarkdown")
library("knitr")
library("sangerseqR")


shinyUI(navbarPage(theme = shinythemes::shinytheme("darkly"),
                   "Sanger Sequence Automatic Analysis Tool",
                   sidebarLayout(
                     sidebarPanel( conditionalPanel(condition =
"!input.example",
                                                    helpText(tags$small("The
file uploaded must be in AB1 Format (*.ab1)")),
                                                    fileInput('files',
'Select AB1 file',
                                                              multiple =
FALSE,
                                                              accept =
c('text/csv', '.ab1'))
                     ),

                     checkboxInput('example', 'Load Example Data', FALSE),


                                         conditionalPanel(condition =
"output.validFile",

h4(tags$span(style="color: blue",'Chromatogram Options')),

                                           h5('End Trimming'),
                                           conditionalPanel(condition =
"!input.autoTrim",

                                                            tags$div(class="float-
left",numericInput("trim5", label = "5'",  value = "0", min = 0,width =
"100")),
                                                            tags$div(class="float-
right",numericInput("trim3", label = "3'",  value = "0", min = 0, width =
"100")),

tags$br(style="clear:both"),
                                                            bsAlert(anchorId =
"long_trim"),
                                                            bsAlert(anchorId =
"na_trim"),

helpText(tags$small("Specify the number of bases to remove from sequence's
ends"))
                                           ),
                                           checkboxInput('aTrm', 'Auto Trim',
FALSE),
                                           helpText(tags$small("Check to perform
an automatic trimming")),
```

```
                                    conditionalPanel(condition =
"output.validFile",

checkboxInput('shwRmBases', 'View Removed Bases', FALSE),

helpText(tags$small("Check to show the bases trimmed from the ends")),

                                                    tags$div(class="float-
left",sliderInput("x", label = "Bases per row", value = "100", min = 10, max
= 400, step = 10)),

tags$br(style="clear:both"),
                                                    bsAlert(anchorId =
"na_width"),

helpText(tags$small("Slide for an appriximative number of bases per row to
display in the chromatogram"))
                                    )
                    ),

                    conditionalPanel(
                      condition = "output.validFile && input.maintabset ==
'Sequence Alignment'",

                      h4(tags$span(style="color: blue",'Sequence Alignment
Option')),

                      h5('Reference Sequence:'),

                      conditionalPanel(condition = "!input.smple",

                                       tags$textarea(id="refseq", rows=8,
cols=50, ""),
                                       fileInput('refFile', 'Select Fasta
file',
                                                 multiple = FALSE,
                                                 accept = c(
                                                   'text/csv',
                                                   '.fasta'),width = '250px'
                                    ) )

                    ), conditionalPanel(condition = "input.smple &&
input.maintabset == 'Sequence Alignment' ",
                                        tags$textarea(id="refsmple",
rows=10, cols=25,

"CATGCGGCGGAGCTCACCGCCGGATTCTACAATGTGGCGGGAAGAGACGGTTATCGGCCTATTGCCAGGATGCTGG
CAAG
GCACCATGCCACTCTGAATTTCACTTGCCTTGAGATGAGAGACTCCGAACAGCCTGCCGAGGCCAAGAGTGCTCCTC
AAG
AACTCGTTCAACAGGTACGTACCATAACCAAAATTCTGCAACATCAAAATGAATATATAAAAGAAGCCTAGGAGCTA
GCA
TAGTGGTTGATCACCCTACCTCAAGTCATAACTGAAATCGGAGGTAGGGGTTGATTGTGTTAGCCTAACTTGAGCAG
TAG
```

```
TGATGTAGTGAATACATCGAAATATACGAATATGTCTAAGTATCGTCTAGGGTTCGGTATGCAATTCAGCTTGAGTA
AAT
GTGCGGGCTTGGGAGCTTTATATAGAGGTTGGAGGGGACACTACCAAAAAAAAGTAGTGCCGCTTTCTTTCATAACA
GAG
TTGTGGTTCCCGGCACATGTCGCGCTTGTCCTTTAGCATAAAAGGATGTGTAATTGAAATAGACATTAAGGTCGGAT
GTT
ATAGAGGTTGGAAGGGACACATGCCGAGCTTACCCTTTAGCATACAAGTATGTCGGATCAAAATAACCCGTCTAGAC
ATT
AAGGTCCGACGTTATAGAGGTTGGAGGGGACACATGTCGGACTTGTGTTTAGCATACAAGGATGTCCGATCAAAATA
ACG
GGTCAGGCGTTCCAACTCAAGTTCAGGGTAAATTGTGCTTCCACTAAGACTCAAATGCACAACCTCCCATATGAAAG
TAC
AGTTTAGTGTCACTAAACCACAAGATCTTAGCTTGTGGTCTTCTTTATTTAAGCCCGGCCACTAGCTATTTACCAAC
TTG
GTTACTTTGGACATGTTAATTAATTAAAGTTGAGTTGAATGATTGTGCAGGTGTTGAGCAGCGGATGGAAAGAGTAT
ATC
GATGTGGCAGGTGAGAATGCACTGCCAAGATATGATGCCACGGCATACAACCAAATGCTTCTAAACGTCAGGCCAAA
CGG
CGTCAACCTTAACGGCCCTCCCAAGCTCAAGATGTCCGGCTTGACATATCTCCGTTTGTCGGACGATCTGTTACAGA
CAG
ACAACTTCGAACTCTTCAAGAAATTCGTCAAGAAGATGCACGCTGATCTG")
                    )
                    ,

                    #helpText(tags$small("Sequence should include at least
the region covered by the sequence results. Non-DNA characters will
automatically be removed.")),
                    conditionalPanel(
                      condition =  "output.validFile && input.maintabset ==
'Sequence Alignment'",
                    bsCollapse(multiple = FALSE, id = "collapse1",
                               bsCollapsePanel("Alignment Type",
                                               radioButtons("align type",
"",
                                                            choices=c(

"Local" = "local",

"Global" = "global",

"Overlap" = "overlap",

"Global-Local" = "global-local",

"Local-Global" = "local-global"),
                                                            selected =
"Local" ))
                    ),

                    helpText(tags$small("Choose the alignment type to be
performed, Local alignment is set by default."))

                    )
                    ),


                    # main panel
```

```
                    mainPanel(

                      tabsetPanel(id="maintabset", selected="How to Use",

                                  tabPanel("DNA Sequence", icon = icon("fas
fa-dna"),

                                           conditionalPanel(
                                             condition =
"!output.validFile",
                                             tags$p("Kindly upload valid
input for sequence information to be displayed.")
                                           ),
                                           conditionalPanel(
                                             condition = "output.validFile",
                                             h5('DNA Sequence:'),
                                             verbatimTextOutput("sqShow"),
                                             uiOutput("dwlF"),
                                             hr(),
                                             h5('AB1 File Details:'),
                                             uiOutput("seqDetails"),
                                             hr(),
                                             h5('Quality Histogram :'),
                                             plotOutput("qHist"),
                                             h5('Cumulative Quality
Histogram:'),
                                             plotOutput("cumHist"),
                                             hr()
                                           )

                                  ),



                                  tabPanel("Chromatogram Viewer",icon =
icon("bar-chart-o"),
                                           conditionalPanel(
                                             condition =
"!output.validFile",
                                             tags$p("Kindly upload valid
input for chromatogram  to be displayed.")
                                           ),
                                           conditionalPanel(
                                             condition = "output.validFile",
                                             tags$p(""),
                                             uiOutput("dlPdf"),
                                             hr(),
                                             plotOutput("chromatogram.view",
height="auto")
                                           )
                                  ),



                                  tabPanel("Sequence Alignment",icon =
icon("fas fa-align-center"), sidebarPanel(
```

```
                                    ),  conditionalPanel(
                                      conditionalPanel(
                                        condition = "!output.refvalidFile",
                                        tags$p("pairwise alignment will be
shown here when a valid reference sequence has been selected.")
                                      ),
                                      conditionalPanel(
                                        condition = "output.refvalidFile",
                                        checkboxInput('rseq', 'Reverse the
main sequence.', FALSE),
                                        helpText(tags$small("check to reverse
the sequence for the alignment.")),
                                        hr(),
                                        h5('Alignment Header:'),
                                        verbatimTextOutput("AHeader"),
                                        h5('Alignment:'),
                                        verbatimTextOutput("AlignView"),
                                        h5('Alignment Program:'),
                                        verbatimTextOutput("APack")
                                      )
                                    )
                                    ),

                                    tabPanel("Polymorphism Detection",icon =
icon("fas fa-eye"),sidebarPanel(


                                    )),

                                    tabPanel("Report",icon = icon("far fa-
clipboard"),
                                             conditionalPanel(
                                               condition =
"!output.validFile",
                                               tags$p("Options to generate a
report will be shown here when a valid input has been uploaded.")
                                             ),
                                             conditionalPanel(
                                               condition = "output.validFile",
                                               uiOutput("generateRptSeq"),
                                               conditionalPanel(
                                                 condition =
"output.refvalidFile",
                                                 uiOutput("generateRptAlign")
                                               ),
                                               radioButtons("rptFormat",
"Report Format:",
                                                            choices=c('PDF',
'HTML', 'Word')),
                                               hr(),
                                               uiOutput("RptFileName"),
                                                 downloadButton('ReportFile',
'Download')
                                             )
                                    ),
```

```
                                    tabPanel("How to Use", icon = icon("fas
fa-chalkboard-teacher"),

                                             includeHTML("howtouse.html")
                                    )









                          )
                        )
                      )
)

)

# End of ui.R
```

Appendix 3: Sample Output the DNA Data In Fasta Format

>BC18_IB2763F

ACGTCGAGCATGCGGCGGAGCTCACCGCCGGATTCTACAATGTGGCGGGAAGAAGACGGT

TATCGGCCTATTGCCAGGATGCTGGAAAGGCACCATGCCACTCTGAACTTCACTTGCCTT

GAGATGAGAGACTCCGAACAGCCTGCCGAGGCCAAGAGTGCTCCTCAAGAACTCGTTCAA

CAGGTACGTACCATAACCAAAATTCTGCAACATCAAAATGAATATATAAAAGAAGCCTAG

GAGCTAGCATAGTGGTTGATCACCCTACCTCAAGTCATAACTGAAATCGGAGGTAGGGGT

TGATTGTGTTAGCCTAATTTGAGCAGTAGTGATGTAGTGAATACATCGAAATATACGAAT

ATGTCTAAGTATCGTCTAGGGTTCGGTATGCAATTCAGCTCGAGTAACTGTGATAGCTTG

CTAGCTAGCTATATATGTAGGATGGAAGACACACTACCAAAAATAATGACTGTCTCTTTC

TTTCCTAACATATGTGTGCCTGCCCGCATACCGCATGTGCTTCACTAGACCTTGATGTGA

AAATGATATATAATCTAAAGACACATGATAGACAGGTTGTAAAGAAGACGTAACGAACTC

ATGCTTAACTAAACCTTTATGATACAACTATGTCACATCTCAATAACTCATCGACACATT

ATAGACAAGCTGTATAGAAGTTGTATGGGACTTGTGTCTAACTTGCGATGATCACCGAAC

GATGTCCCATCTCAAGCACTTCCACCTGCATTCCAACTCAAATTGAGGGTACATTGAGAT

TCCACTGCGACTCCTCTGCATATGCCTCTCCTATTTAAGTGTCACTAAACTGTCACATCT

TAACATGATCTCGTCTTGATCTTCTCTATGCCAGCTAACTATCTACCTACCTGCTTGCTT

ACTACTGACAGTTAATTAAAGTTGAGTTGAATGATTGTGCAGTGTAGAGAGAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAATATTTAAAAAAAAAAACCCCAAGAAGGAA

GGAAGGAAGGAAGGGGGGGAAGAAAAAGAGGGAGAGGAAAGAGAGAAGAAGAGGGGAAGA

GAGGGGAGGAAGGGGGGAGGGGAAGAAGGAGAAGGAAGAAGGAGAGGGGAGAAGAAGGAG

AAGGGAAGAAGAGAGAAAGAAGGAAAGAGAAGGGGAAAGGGGGGGGAGGAGAAGAGAGAG

AGGAGGGAAGAAGGAGAAGAGGAGAAAAGAAAGAAGGGGGAGAGAAGGAAGAGAAGGGAG

AGGGGAGGGAGAAGAAGAAGAAGAGGAAAGAGAAAGGAAGGGGGAAAAGGGAGAGAGGAG

AGAAGAGAGAGGAGAGAGAGAAAAGAGAAAGAAGGAGAAGAAGAAGAGGGAGGAGAG

GAGGAAAGAAGAAAAGAGAGAGGGAGAGAAGAGGAAGAAGGGGAGGAGGGAGAAGGAAGA

GGGAAAAGAAGGAGAGAAAAGAAAAAGAGAAAAGAGAGAGGAAGAGAGGAAGAAGAGAGA

AAAGGGAGAAGAGGAGACGAAAAAGGAAGAGAAAGGGAAGAAAGAGGAGGAGGAGGAGAA

AAGAAGAAGGAAAAGAAGGAGGAAGAAGGAAGAGGGAGAGGGAGAAAGGAAAGAAAAGAG

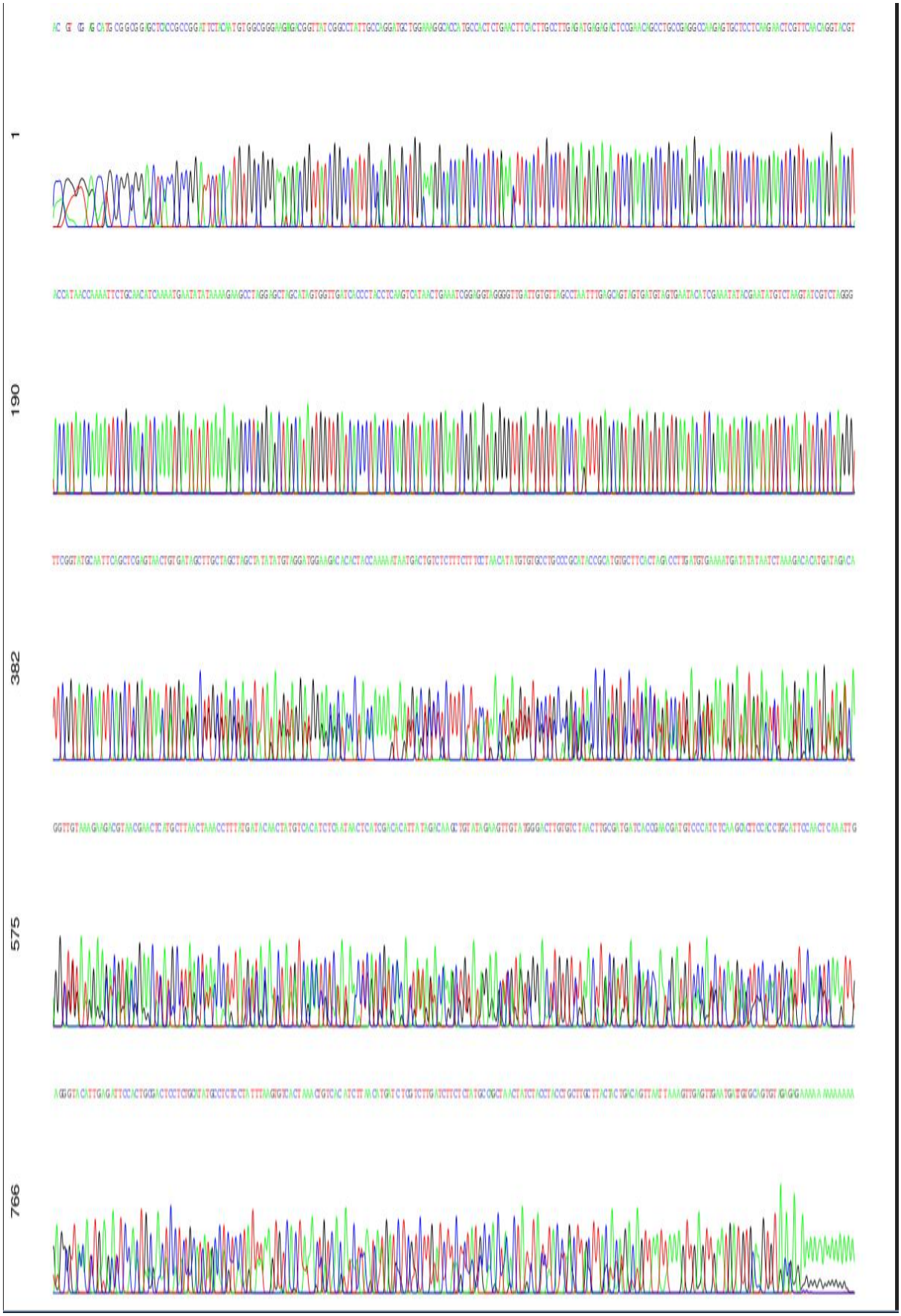
GGAAAAGAGAGAGGAGAGAAAGAGAAAG

Appendix 4: Sequence Alignment Results

```
Aligned_sequences: 2
1: P1
2: S1
Matrix: NA
Gap_penalty: 14.0
Extend_penalty: 4.0

Length: 404
Identity:     398/404 (98.5%)
Similarity:    NA/404 (NA%)
Gaps:           1/404 (0.2%)
Score: 745.2407
```

```
P1                9 CATGCGGCGGAGCTCACCGCCGGATTCTACAATGTGGCGGGAAGAAGACGGTTATCGGCC     68
                    |||||||||||||||||||||||||||||||||||||||||||||| ||||||||||||||
S1                1 CATGCGGCGGAGCTCACCGCCGGATTCTACAATGTGGCGGGAAGA-GACGGTTATCGGCC     59

P1               69 TATTGCCAGGATGCTGGAAAGGCACCATGCCACTCTGAACTTCACTTGCCTTGAGATGAG    128
                    |||||||||||||||| |||||||||||||||||||| ||||||||||||||||||||
S1               60 TATTGCCAGGATGCTGGCAAGGCACCATGCCACTCTGAATTTCACTTGCCTTGAGATGAG    119

P1              129 AGACTCCGAACAGCCTGCCGAGGCCAAGAGTGCTCCTCAAGAACTCGTTCAACAGGTACG    188
                    ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
S1              120 AGACTCCGAACAGCCTGCCGAGGCCAAGAGTGCTCCTCAAGAACTCGTTCAACAGGTACG    179

P1              189 TACCATAACCAAAATTCTGCAACATCAAAATGAATATATAAAAGAAGCCTAGGAGCTAGC    248
                    ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
S1              180 TACCATAACCAAAATTCTGCAACATCAAAATGAATATATAAAAGAAGCCTAGGAGCTAGC    239

P1              249 ATAGTGGTTGATCACCCTACCTCAAGTCATAACTGAAATCGGAGGTAGGGGTTGATTGTG    308
                    ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
S1              240 ATAGTGGTTGATCACCCTACCTCAAGTCATAACTGAAATCGGAGGTAGGGGTTGATTGTG    299

P1              309 TTAGCCTAATTTGAGCAGTAGTGATGTAGTGAATACATCGAAATATACGAATATGTCTAA    368
                    |||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||
S1              300 TTAGCCTAACTTGAGCAGTAGTGATGTAGTGAATACATCGAAATATACGAATATGTCTAA    359

P1              369 GTATCGTCTAGGGTTCGGTATGCAATTCAGCTCGAGTAACTGTG    412
                    |||||||||||||||||||||||||||||| |||||| ||||
S1              360 GTATCGTCTAGGGTTCGGTATGCAATTCAGCTTGAGTAAATGTG    403
```

Appendix 5: Chromatogram Viewer Results


1
190
382
575
766

RESEARCH OUTPUTS

Publication Paper

Mero, V., & Machuve, D. (2021). The Usability Testing of SSAAT, a Bioinformatic Web Application for DNA Analysis at a Nucleotide Level. *Engineering, Technology & Applied Science Research*, *11*(3), 7075-7078.

Poster Presentation