

OBJECT-ORIENTED DESIGN AND UI PROTOTYPING FOR A BIOINFORMATICS TOOL TO DETECT AND ANNOTATE NGS VARIANTS OF PLASMODIUM FALCIPARUM IN INDIVIDUALS WITH SICKLE CELL DISEASE.

Abstract:

The coevolution between sickle cell disease and malaria illustrates a major mechanism of natural selection in which the hemoglobin S allele confers protection against severe forms of infection. In HbAS and HbSS carriers, however, the altered red-cell milieu also pressures *P. falciparum* itself and can favor particular parasite genotypes. Next-generation sequencing (NGS) is particularly well-suited for monitoring this interaction, as it allows for the identification of parasite mutations and potential adaptive changes with high precision. In practice, libraries derived from blood consist mainly of human DNA; therefore, to obtain reliable results regarding the parasite, it is necessary to use processes that remove host-related background noise beforehand. This article presents the object-oriented design approach and the prototyping of a human-machine interface developed using the Unified Modeling Language (UML) as part of the DREPAL project (sickle cell disease and malaria). The work aims to structure the IPCINGSTOOLKIT platform for in silico host decontamination, variant detection, and annotation, thereby contributing to comparative analysis of *Plasmodium falciparum* genomes and to the identification of functional adaptations according to sickle cell genotype. The characterization of actors involved in the analyses, modeling through use case, activity, and deployment diagrams, and the interface mockups designed prior to software implementation are detailed in this work.

Key words:-

Object-Oriented Design, UI Prototyping, UML, NGS, Plasmodium falciparum, Sickle Cell Disease, Human Decontamination, Variant Calling, Annotation.

Introduction:-

The coevolution between the human host with sickle cell disease and the parasite *Plasmodium falciparum* is one of the most remarkable examples of natural selection in humans[1–4]. Although the presence of the hemoglobin S allele in the heterozygous state (HbAS) confers well-documented protection against severe malaria, the impact of sickle cell genotypes (HbAS and HbSS) on the diversity and adaptation of the parasite genome remains a critical research field [5, 6].

The advent of next-generation sequencing (NGS) makes it possible to explore this genetic variability at high resolution[7–9]. Nevertheless, bioinformatics analysis of parasite isolates collected from patients faces two major challenges:

- Host contamination: a predominant proportion of raw sequencing data from whole blood corresponds to the human genome, requiring a strict decontamination step and exclusive extraction of *Plasmodium falciparum* sequences.
- Multi-group comparison and annotation: the identification and characterization of significant variants (SNPs and INDELs) reflecting functional adaptation in HbAS and HbSS subjects relative to HbAA controls.

However, working with these complex bioinformatics pipelines often faces the technical barrier of command-line interfaces, which remain difficult for biologists and clinicians to use[10–12]. To overcome this constraint, modern software engineering advocates an interdisciplinary approach that combines rigorous system modeling with user-

centered design[13]. Explicit formalization of data flows and simplification of human–computer interaction thus become essential conditions to ensure analysis traceability, reduce execution errors, and enable rapid uptake of tools by medical and scientific teams[14].

To meet these needs within the DREPAL project (Sickle Cell Disease–Malaria), carried by the Parasitology–Mycology Unit of the Pasteur Institute of Côte d’Ivoire (IPCI), the development of an integrated and ergonomic bioinformatics application, “IPCINGSTOOLKIT,” is required. The success of such a tool rests on a prior stage of rigorous software engineering. This article describes the object-oriented design approach based on the UML (Unified Modeling Language) and the prototyping of graphical user interfaces (GUIs), ensuring alignment between researchers’ needs and the future software implementation[15, 16].

Materials and Methods:-

Project Scope and Data Specifications

The software is designed to process complete Illumina next-generation sequencing (NGS) data from *Plasmodium falciparum* isolates collected from three cohorts of subjects characterized by their phenotype and hemoglobin type:

- healthy control subjects (HbAA);
- subjects with the sickle cell trait (HbAS);
- subjects with major sickle cell disease (HbSS).

Host and parasite DNA are recovered together from whole blood; to keep each analysis step specific, the design therefore relies on two standard reference genomes:

- Human reference (GRCh38 / hg38): used in the in silico decontamination step to map and discard reads of human origin.
- Pathogen reference (*Plasmodium falciparum* 3D7): used after host removal to align retained parasite reads, call SNPs/INDELs, and annotate predicted effects on parasite coding regions.

Software Engineering Methodology

The Unified Modeling Language (UML) was used to structure the object-oriented analysis and design of the “IPCINGSTOOLKIT” framework by translating the requirements of the DREPAL project into rigorous technical specifications. This formalization of the software architecture is organized into two main stages, illustrated by the flowchart in Figure 1, for which Draw.io (<https://app.diagrams.net/>) was selected as the single environment for producing the chain of graphical artifacts.

•Requirements analysis

Requirements analysis (Figure 1-A) was structured into four methodological steps: gathering user needs, prioritizing requirements, validating requirements, and then specifying them. This process, supported by a traceability framework, ensured the conformity, consistency, and reliability of the framework during its implementation.

•Conceptual modeling of requirements

The modeling of IPCINGSTOOLKIT (Figure 1-B) relies on three UML diagrams: the use case diagram [17] defines actor interactions with the system; the activity diagram formalizes the dynamics and logic of business workflows [18]; and the deployment diagram [18] maps the physical architecture and the integration of software components onto the high-performance computing (HPC) analysis infrastructure.

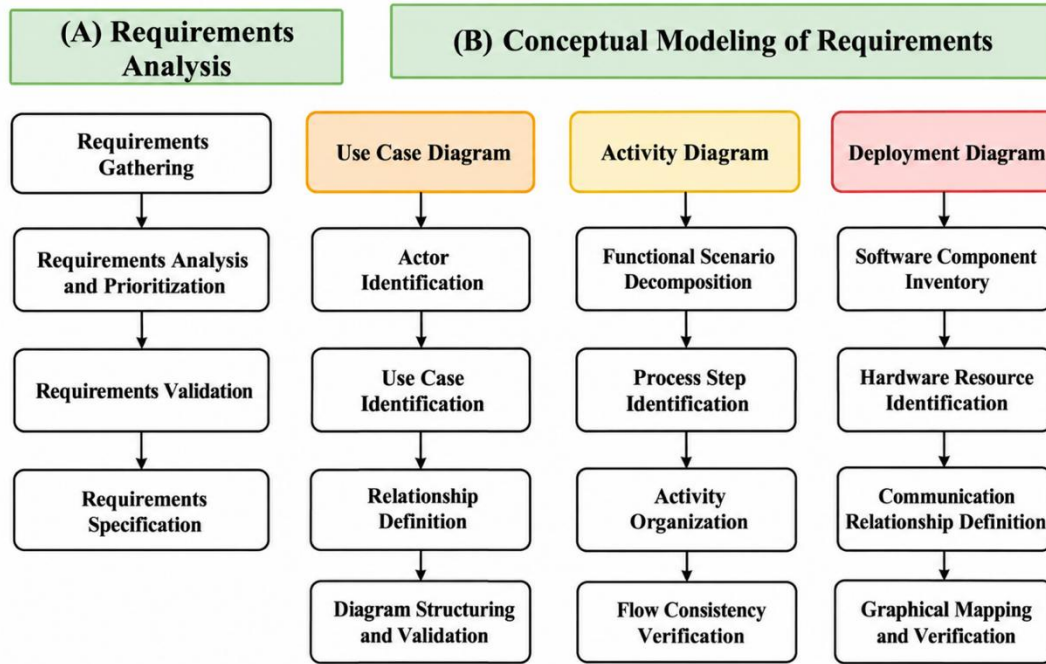


Figure 1:- Object-oriented design process for the IPCINGSTOOLKIT tool.
(A) Requirements analysis. (B) Conceptual system modeling

GUI Prototyping

Graphical user interface (GUI) prototyping is a key step in software engineering, consisting in designing a high-fidelity visual and interactive prototype before coding begins. This approach aims to define precisely the ergonomics, the spatial layout of graphical components (import areas, comparison filters, interactive tables, and annotation modules), and user journeys (UX). By faithfully modeling the visual environment, this phase makes it possible to validate functional specifications in advance with researchers and clinicians, to optimize accessibility to complex bioinformatics features, and to substantially reduce code-rewriting costs during subsequent development.

Results:-

Actor and Use Case Analysis

Our conceptual model distinguishes two main categories of actors, as listed in Table 1. The first category comprises primary actors, i.e., users directly involved in interacting with the developed system. These include the bioinformatician (BI), responsible for setting up and executing bioinformatics analyses (data preprocessing, variant detection and annotation, SNP matrix generation, etc.), and the biologist (Bio), responsible for the biological interpretation of results produced within the DREPAL project. The latter relies on the system's analytical outputs to explore the adaptive mechanisms of *Plasmodium falciparum* in the sickle cell context.

Table 1:-List of IPCINGSTOOLKIT actors

Category	Actor	Roles
Primary	Bioinformatician (BI)	This profile is key to ensuring the robustness of analyses and validating the quality of bioinformatics results.
	Biologist (Bio)	The biologist uses a simplified interface dedicated to data analysis for annotation validation, characterization of functional variants, and biological interpretation.
Secondary	Administrator(ADM)	This role ensures access security, reliability, and continuity of system service.

The architecture of IPCINGSTOOLKIT relies on interdisciplinary complementarity in which primary actors (the bioinformatician, who guarantees technical quality, and the biologist, who guides exploitation of the results) execute the analytical use cases of the pipeline, among others, sequence loading, quality control, in silico double subtraction (host–parasite), variant calling, and annotation, while the administrator, as a secondary actor, provides overall supervision, configuration, and access control to ensure system security and maintenance. All users share a transversal authentication component that secures access to the platform (Table 2).

Table 2:-List of IPCINGSTOOLKIT use cases (UC)

Category	Use case	BI	Bio	ADM
Generic	Authentication	x	x	x
	LoadSequences	x	x	
	Qualitycontrol	x	x	
	In silico double subtraction	x	x	
	Variant calling /Annotation	x	x	
Supervision	Configure IPCINGSTOOLKIT	x		x
	Maintain IPCINGSTOOLKIT	x		x
	Manage user			x

Use-case diagram

Figure 2 presents the use-case diagram (UCD) of IPCINGSTOOLKIT. It shows which actions each user profile—bioinformatician, biologist or administrator—can launch on the platform. Casting the pipeline as separate use cases makes responsibilities easier to audit and provides a checklist against which planned functions can be matched to real laboratory needs.

Four main use cases organize the diagram:

- Sequence and reference loading: upload of raw reads together with the host and pathogen reference genomes required downstream.
- Quality control: review of sequencing metrics, then trimming and filtering of low-quality reads. The related sub-cases are “visualize sequence quality” and “trim and filter sequences.”
- In silico double subtraction: first drop human-origin reads, then keep reads that match the target parasite. This use case relies on the sub-cases “host subtraction” and “reference indexing.”
- Variant calling and annotation: identification of polymorphisms—particularly SNPs—on purified parasite sequences, and association of functional information (protein impact, resistance mechanisms). This use case involves the sub-cases “variant detection,” “variant filtering,” and “annotation of filtered variants.”

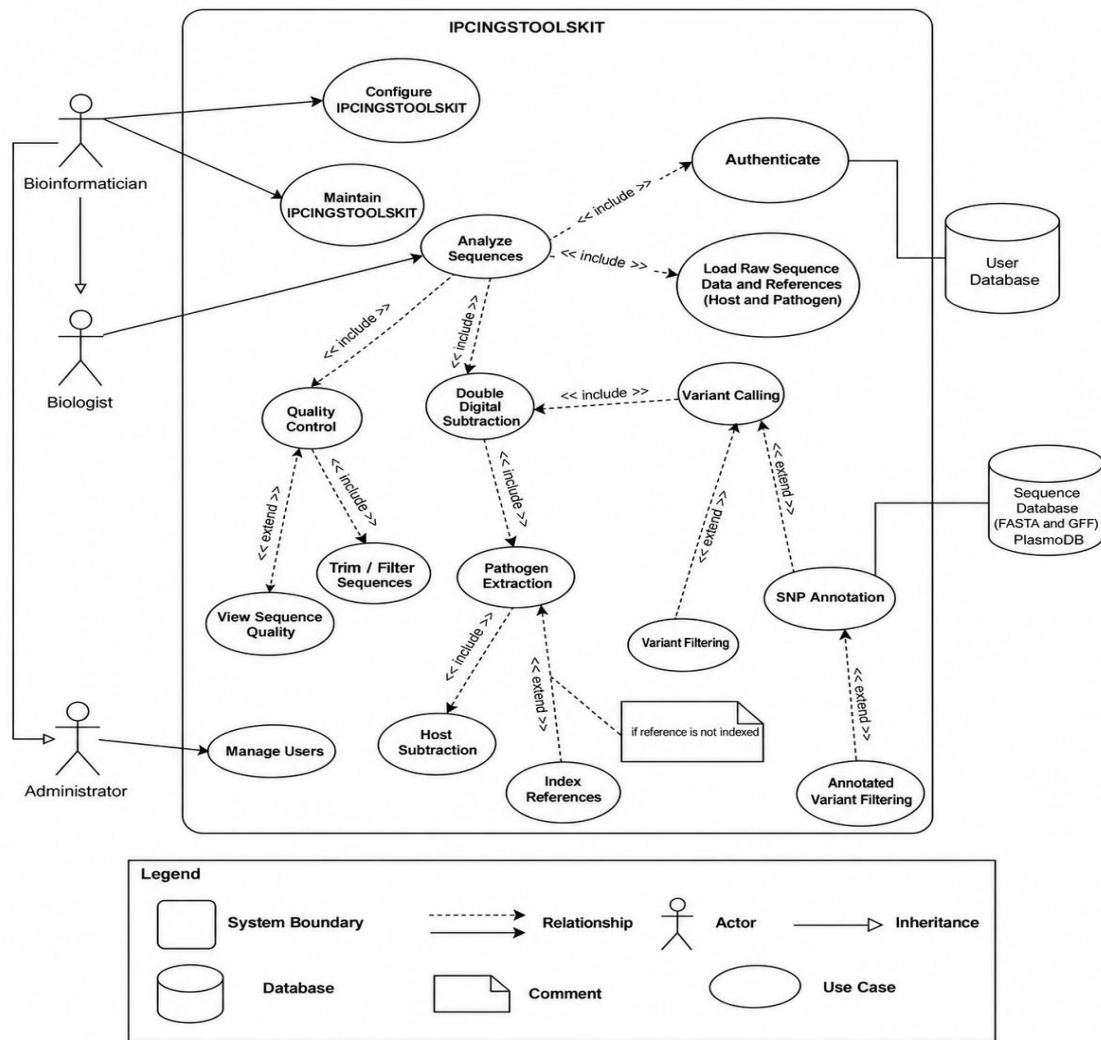


Figure 2:-Use case diagram of IPCINGSTOOLKIT

Activity diagrams for the IPCINGSTOOLKIT workflows

Activity diagram for the parasite sequence extraction process

The activity diagram in Figure 3 models the automated process of the IPCINGSTOOLKIT pipeline for isolating *Plasmodium falciparum* sequences from raw NGS data. This workflow comprises three main steps:

- Step 1: Quality control of NGS reads evaluates quality parameters (length, Phred scores, ambiguous bases) and removes low-quality sequences to produce cleaned data free of technical noise via the FastQC [19], MultiQC [20], and Trim Galore [21] tool suite.
- Step 2: Alignment to the human genome and removal of host sequences aligns the cleaned reads to the human reference genome (GRCh38) in order to extract and retain only unmapped sequences (“non-human sequences”).
- Step 3: Targeted extraction of parasite sequences finalizes the specific selection of reads belonging to the parasite for downstream analyses.

The last two steps rely on BWA [22] and SAMtools [23].

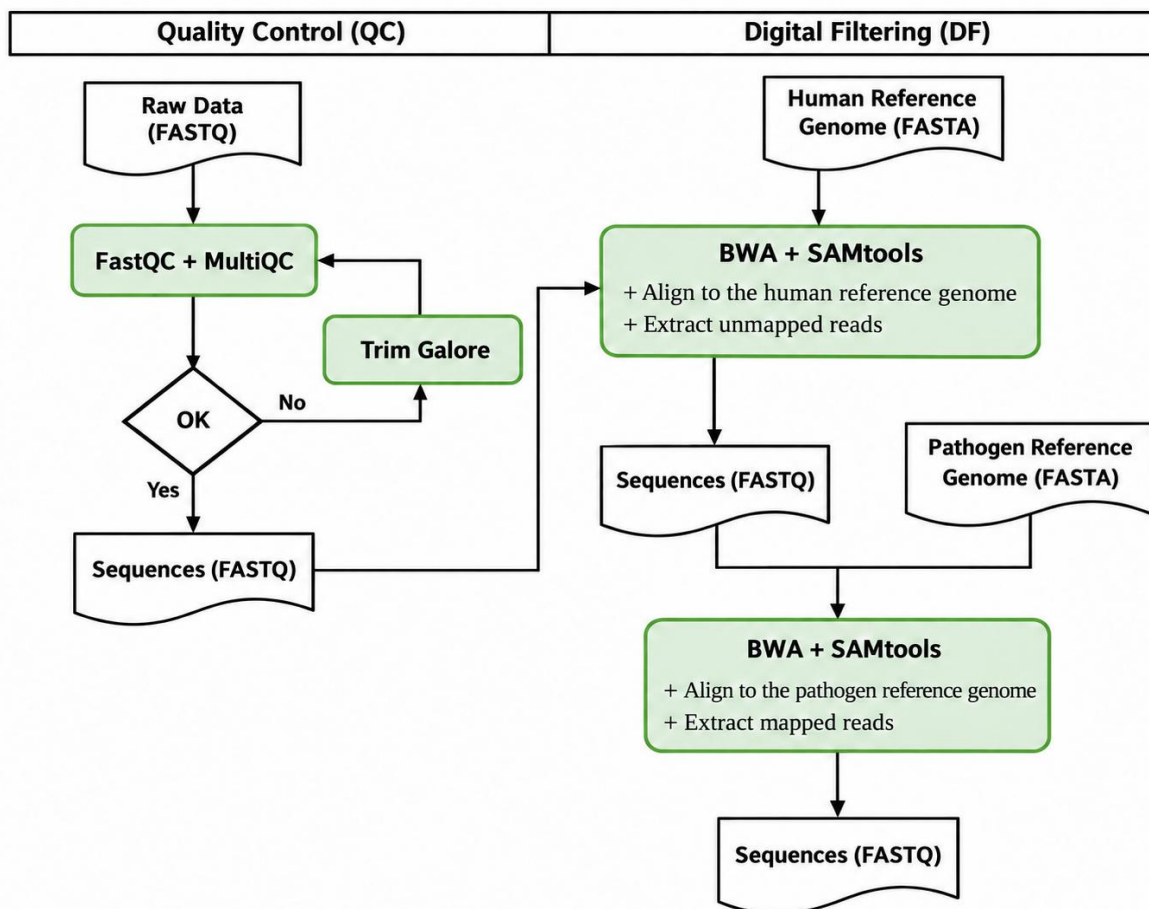


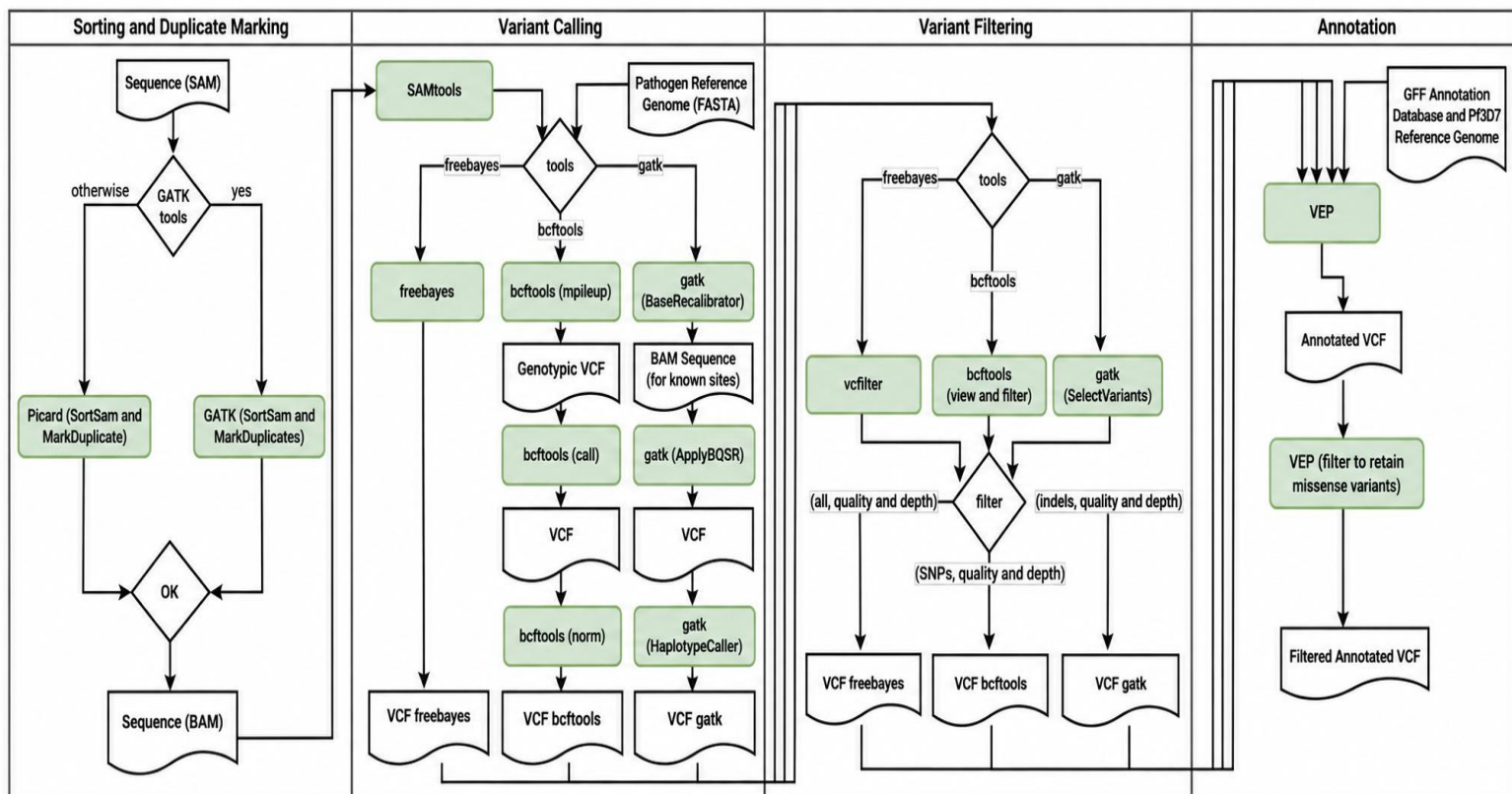
Figure 3:- Activity diagram of the parasite sequence extraction process in IPCINGSTOOLKIT

Activity diagram of the variant detection and annotation process for parasite sequences

Figure 4 presents the activity diagram for detection and annotation of genetic variants from the extracted parasite reads, structured into four key steps:

- Step 1: Sorting and duplicate marking; aligned reads are sorted and PCR duplicates are identified using Picard (MarkDuplicates) (<http://broadinstitute.github.io/picard/>) or GATK (MarkDuplicates) [24] in order to eliminate biases during mutation calling.
- Step 2: Mutation calling (variant calling); the user selects two tools among three available options (GATK HaplotypeCaller [24], BCFtoolsmpileup/call [25], or FreeBayes [26]). Only variants in the intersection (shared by the two selected tools) are retained to maximize reliability.

Figure 4:- Activity diagram of the genetic variant detection and annotation process in IPCINGSTOOLKIT



•Step 3: Variant filtering; raw variants from the VCF are filtered by type, depth (e.g., $DP > 10$), and quality (e.g., $QUAL > 20$) using the tool adapted to the retained variant caller (vcfilter, bcftools filter, or GATK SelectVariants + VariantFiltration).

•Step 4: Variant annotation; filtered variants are functionally annotated with Ensembl VEP (Variant Effect Predictor) [27], which associates each variant with information on its transcript consequence, protein impact, and potential resistance mechanisms.

Physical Architecture and Deployment Diagram

The deployment diagram (Figure 5) formalizes the distributed client-server physical architecture of the IPCINGSTOOLKIT platform, specifically dimensioned to meet the highly intensive requirements of NGS genomic data processing.



Figure 5:- Deployment diagram of IPCINGSTOOLKIT

This modeling describes the hardware topology and the strategic distribution of software components, specifying execution nodes, network communication flows, and the computing resources allocated to each pipeline step through two complementary functional nodes:

- The user client workstation (frontend): dedicated to user interaction through an interface built with Python and Streamlit (<https://streamlit.io/>), it enables data loading, analysis parameterization, result visualization, and report consultation in a fluid and intuitive manner.
- The application server (backend) executes the heavy processing steps: quality control, host–parasite decontamination, variant detection via GATK, FreeBayes, and BCFtools, followed by annotation with VEP. These processes are orchestrated by Python and Bash scripts and rely on the Bioconda package manager [28] to ensure portability and reproducibility of analyses. Such a modular architecture enables secure, high-performance, and scalable remote execution of the IPCINGSTOOLKIT pipeline for processing parasite NGS data.

Graphical User Interface (GUI) Prototyping

IPCINGSTOOLKIT home screen

The interface prototype represented by the IPCINGSTOOLKIT home screen (Figure 6) constitutes the main entry point of the system, providing an ergonomic interface and hierarchical menus to guide bioinformaticians, researchers, and administrators. This functional mockup centralizes access to all analytical steps (from data loading and quality control through decontamination, calling, annotation, and visualization of variants) while integrating contextual assistance tools to facilitate use in a multidisciplinary setting.

Figure6:- IPCINGSTOOLKIT home screen

Prototypes of the secondary menus of IPCINGSTOOLKIT

Prototyping of the IPCINGSTOOLKIT platform is enriched by specialized interfaces (Figure 7) dedicated to key pipeline steps, such as quality control, reference management, or variant annotation. These functional mockups enable the user to drive processing algorithms while ensuring traceability and reproducibility of analyses within a fluid and accessible environment.

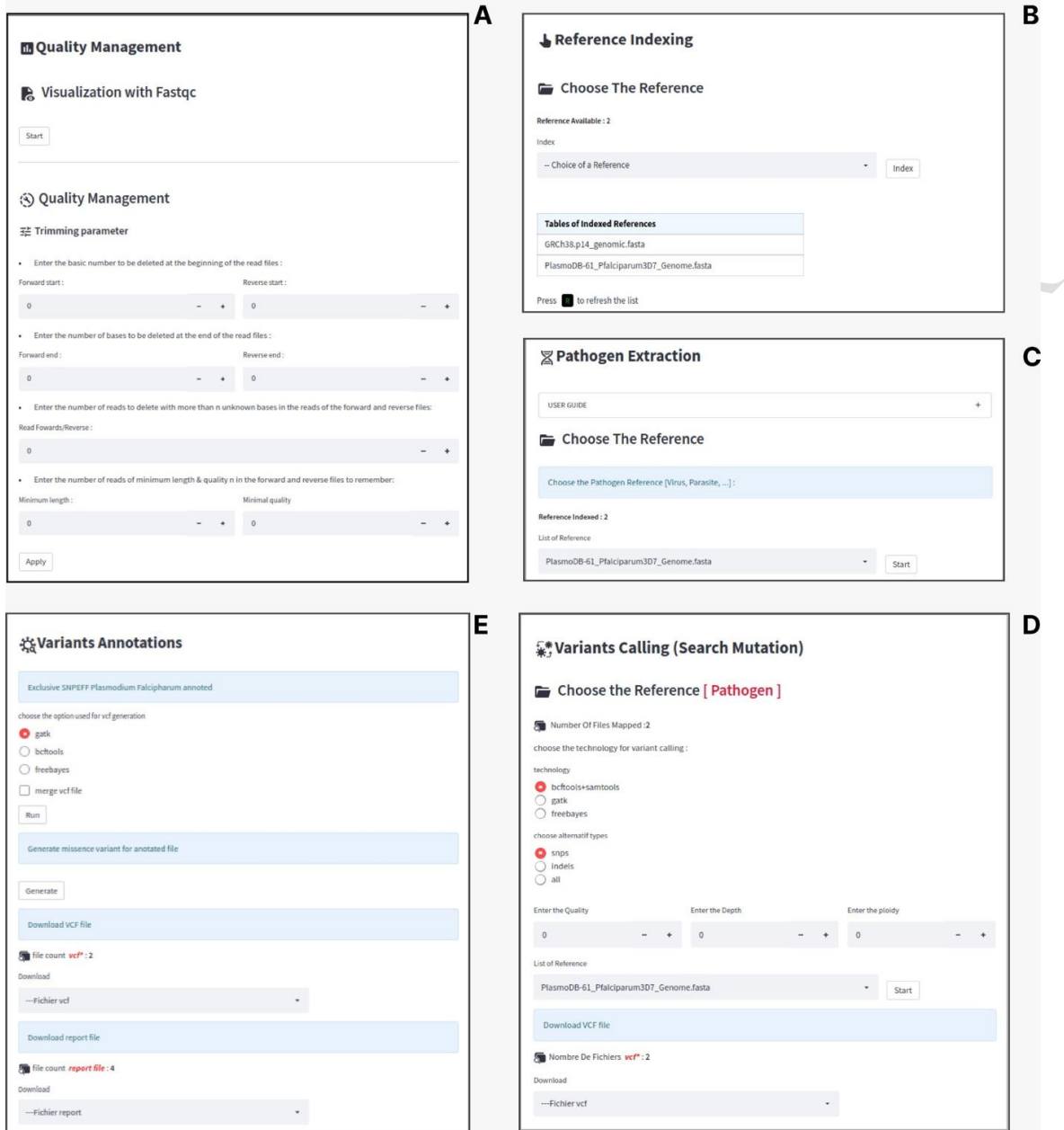


Figure 7:- Secondary interfaces and detailed views of IPCINGSTOOLKIT. (A) Quality management. (B) Indexing of reference genomes. (C) Extraction of parasite sequences. (D) Variant calling. (E) Variant annotation.

- Sequence quality-control prototype (Figure 7-A): interface dedicated to the evaluation of sequencing metrics and to trimming of low-quality reads.
- Reference indexing screen (Figure 7-B): register and index host and *P. falciparum* FASTA files before alignment.
- Parasite extraction screen (Figure 7-C): apply host subtraction to retain *P. falciparum* reads; main parameters remain user-adjustable.
- Variant calling screen (Figure 7-D): run GATK, FreeBayes and/or BCFtools, apply quality filters, and compute simple statistics to highlight informative SNPs.

•Variant annotation prototype (Figure 7-E): biological interpretation mockup relying on VEP to assess the functional impact of mutations on target genes and proteins.

Discussion:-

The IPCINGSTOOLKIT platform addresses software-engineering challenges in medical bioinformatics by combining system modeling with UML and user-centered design based on GUI prototyping.

UML modeling structures the complexity of NGS pipelines by abstracting their high-volume flows and algorithmic sequences at three complementary levels. Use-case diagrams state who may perform which task and which functions sit inside the system boundary. Activity diagrams then list the processing order, quality control, host decontamination, variant calling and VEP annotation, so each run can be traced. The deployment diagram places the light client apart from the containerized nodes that carry out the heavy computation.

Working with UML views and GUI mockups in parallel brings biologists and developers onto the same screens before coding starts, which limits late mismatches between expected and delivered behavior. Scripts remain in the background, but users still set the main analysis parameters. On-screen plots and short contextual tips are included to reduce operator errors when *P. falciparum* variants are inspected.

The complementarity between UML diagrams and GUI prototypes ensures close alignment between software structure and the user's visual perception, guaranteeing maintainability and evolvability of the system.

Positioned against the scientific literature on software engineering applied to computational biology, this approach fully aligns with current standards:

- Model-Driven Engineering (MDE): numerous works in the field (notably formalizations of workflows under Taverna, Galaxy, or Nextflow described in the bioinformatics engineering literature) advocate prior abstract modeling. The use of UML in IPCINGSTOOLKIT joins rigorous approaches applied to health genomics platforms, where activity diagrams help formalize scientific data provenance [29, 30].

- Reducing the “bioinformatics / biologist” gap: in the design of web bioinformatics platforms for variant annotation (such as cBioPortal [31], Galaxy [32], or proprietary clinical diagnostic pipelines), recent work emphasizes the need to couple object-oriented architecture with early UX/UI prototyping. Using a unified approach to link object-oriented modeling and GUI mockups avoids common pitfalls of academic bioinformatics software, often criticized for poor ergonomics, by ensuring immediate alignment between biological concepts and software structure [13, 33, 34].

- Distributed modular architecture: as with modern medical bioinformatics frameworks that rely on UML to define encapsulation in containers (Docker/Singularity), the IPCINGSTOOLKIT deployment diagram ensures a clear separation between the presentation layer (frontend) and the execution layer (analytical backend), thereby ensuring portability and reusability of the system across different computing infrastructures [35, 36].

Conclusion and Perspectives:-

This article has formalized the object-oriented design and GUI prototyping of a bioinformatics application dedicated to NGS variant analysis of *Plasmodium falciparum* within the DREPAL project. The use of UML and prototyping tools made it possible to structure the entire system with precision, from the initial decontamination of human sequences through to the annotation of SNPs of functional adaptation in sickle cell subjects.

As perspectives, future work will focus on the effective implementation of the pipeline as an automated and reproducible workflow developed in Snakemake [37] or Nextflow [38], coupled with complete development of the web front end based on the validated functional mockups. These developments will further be enriched by the integration of machine-learning models to predict the impact of identified parasite variants on the clinical severity of sickle cell crises.

Acknowledgments:-

The authors would like to express their gratitude to the DREPAL project team and its partners for the scientific expertise they provided. They also thank their collaborators at the Pasteur Institute of Côte d'Ivoire (IPCI), Jean LorougnonGuédé University (UJLoG), Félix Houphouët-Boigny University (UFHB), and the National Malaria Control Program (PNLP) for their methodological and technical support.

Funding:-

The authors thank the Rotary Foundation for its financial support, which made this study possible.

Conflict of Interest Statement:-

The authors declare no conflict of interest.

Author Contributions:-

The authors contributed to the study as follows: AES conceived and coordinated the project and performed the UML modeling. AES, KBM, KKAJ, AAAB, and SCR supervised the methodological design and validated the results; AES, KBM, and KKAJ drafted the manuscript with contributions from all authors. AES, AAAB, DAJ, AKJ, and JR read and approved the final version of the manuscript.

References:-

- [1] E. Elguero et al., "Malaria continues to select for sickle cell trait in Central Africa," *Proceedings of the National Academy of Sciences*, vol. 112, no. 22, pp. 7051–7054, Jun. 2015, doi: 10.1073/pnas.1505665112.
- [2] G. Band et al., "Malaria protection due to sickle haemoglobin depends on parasite genotype," *Nature*, vol. 602, no. 7895, pp. 106–111, Feb. 2022, doi: 10.1038/s41586-021-04288-3.
- [3] K. Matuschewski and A. G. Maier, "Pas-de-deux: African *Plasmodium falciparum* adaptations to sickle hemoglobin," *Trends in Parasitology*, vol. 38, no. 4, pp. 267–268, Apr. 2022, doi: 10.1016/j.pt.2022.01.009.
- [4] T. N. Williams et al., "Sickle Cell Trait and the Risk of *Plasmodium falciparum* Malaria and Other Childhood Diseases," *The Journal of Infectious Diseases*, vol. 192, no. 1, pp. 178–186, Jul. 2005, doi: 10.1086/430744.
- [5] E. Gómez-Díaz and L. Ranford-Cartwright, "Evolutionary race: Malaria evolves to evade sickle cell protection," *Cell Host & Microbe*, vol. 30, no. 2, pp. 139–141, Feb. 2022, doi: 10.1016/j.chom.2022.01.010.
- [6] F. Ntoumi, C. Rogier, A. Dieye, J.-F. Trape, P. Millet, and O. Mercereau-Puijalon, "Imbalanced Distribution of *Plasmodium falciparum* MSP-1 Genotypes Related to Sickle-Cell Trait," *Molecular Medicine*, vol. 3, no. 9, pp. 581–592, Sep. 1997, doi: 10.1007/BF03401815.
- [7] N. Billows et al., "Global-scale population genetic analysis of *Plasmodium falciparum* identifies region-specific patterns of malaria parasite adaptation," *Nature Communications*, vol. 17, no. 1, p. 6318, May 2026, doi: 10.1038/s41467-026-73006-2.
- [8] J. Zerebinski and D. F. Plaza, "Progress in malaria genomic surveillance using long-read sequencing," *Trends in Parasitology*, vol. 40, no. 11, pp. 956–958, Nov. 2024, doi: 10.1016/j.pt.2024.10.001.
- [9] A. Adekile, N. Akbulut-Jeradi, R. Al Khaldi, M. J. Fernandez, and J. Sukumaran, "Diagnosis of Sickle Cell Disease and HBB Haplotyping in the Era of Personalized Medicine: Role of Next Generation Sequencing," *Journal of Personalized Medicine*, vol. 11, no. 6, p. 454, Jun. 2021, doi: 10.3390/jpm11060454.
- [10] J. Rosa, J. Andrade, J. M. Silva, and J. L. Oliveira, "Biochef: a client-side WebAssembly-based workflow builder for genomic data analysis," *BMC Bioinformatics*, vol. 27, no. 1, p. 103, Apr. 2026, doi: 10.1186/s12859-026-06431-1.
- [11] G. Lalli, J. Collier, Y. Moreau, and D. Raimondi, "JINet: easy and secure private data analysis for everyone," *BMC Bioinformatics*, vol. 26, no. 1, p. 248, Oct. 2025, doi: 10.1186/s12859-025-06244-8.

- [12] A. Bimbocci, M. Baragli, and A. Magi, “SnakeBITE: A SNAKEmake-Based Interface for Third-Generation Sequencing Data Analysis,” *Biomolecules*, vol. 16, no. 2, p. 314, Feb. 2026, doi: 10.3390/biom16020314.
- [13] K. Pavelin, J. A. Cham, P. de Matos, C. Brooksbank, G. Cameron, and C. Steinbeck, “Bioinformatics Meets User-Centred Design: A Perspective,” *PLOS Computational Biology*, vol. 8, no. 7, p. e1002554, Jul. 2012, doi: 10.1371/journal.pcbi.1002554.
- [14] A. A. Bayor, J. Li, I. A. Yang, and M. Varnfield, “Designing Clinical Decision Support Systems (CDSS)—A User-Centered Lens of the Design Characteristics, Challenges, and Implications: Systematic Review,” *Journal of Medical Internet Research*, vol. 27, no. 1, p. e63733, Jun. 2025, doi: 10.2196/63733.
- [15] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, 2nd ed. Upper Saddle River, NJ: Addison-Wesley, 2005.
- [16] A. Azadi and F. J. García-Peñalvo, “Optimizing Clinical Decision Support System Functionality by Leveraging Specific Human-Computer Interaction Elements: Insights From a Systematic Review,” *JMIR Human Factors*, vol. 12, p. e69333, May 2025, doi: 10.2196/69333.
- [17] I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard, *Object-Oriented Software Engineering: A Use Case Driven Approach*. Reading, MA: Addison-Wesley, 1992.
- [18] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*, 2nd ed. Pearson Higher Education, 2004.
- [19] S. Andrews, “FastQC: A Quality Control Tool for High Throughput Sequence Data,” Cambridge: Babraham Bioinformatics, 2010.
- [20] P. Ewels, M. Magnusson, S. Lundin, and M. Käller, “MultiQC: summarize analysis results for multiple tools and samples in a single report,” *Bioinformatics*, vol. 32, no. 19, pp. 3047–3048, Oct. 2016, doi: 10.1093/bioinformatics/btw354.
- [21] F. Krueger, F. James, P. Ewels, E. Afyounian, and B. Schuster-Boeckler, “FelixKrueger/TrimGalore: v0.6.7 - DOI via Zenodo,” Zenodo, Jul. 23, 2021, doi: 10.5281/ZENODO.5127899.
- [22] H. Li and R. Durbin, “Fast and accurate short read alignment with Burrows–Wheeler transform,” *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, Jul. 2009, doi: 10.1093/bioinformatics/btp324.
- [23] H. Li et al., “The Sequence Alignment/Map format and SAMtools,” *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, Aug. 2009, doi: 10.1093/bioinformatics/btp352.
- [24] A. McKenna et al., “The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data,” *Genome Research*, vol. 20, no. 9, pp. 1297–1303, Sep. 2010, doi: 10.1101/gr.107524.110.
- [25] P. Danecek et al., “Twelve years of SAMtools and BCFtools,” *GigaScience*, vol. 10, no. 2, p. giab008, Jan. 2021, doi: 10.1093/gigascience/giab008.
- [26] E. Garrison and G. Marth, “Haplotype-based variant detection from short-read sequencing,” arXiv, 2012, doi: 10.48550/ARXIV.1207.3907.
- [27] F. Cunningham et al., “Ensembl 2022,” *Nucleic Acids Research*, vol. 50, no. D1, pp. D988–D995, Jan. 2022, doi: 10.1093/nar/gkab1049.
- [28] The Bioconda Team et al., “Bioconda: sustainable and comprehensive software distribution for the life sciences,” *Nature Methods*, vol. 15, no. 7, pp. 475–476, Jul. 2018, doi: 10.1038/s41592-018-0046-7.
- [29] Q. Yan, “Bioinformatics for Transporter Pharmacogenomics and Systems Biology: Data Integration and Modeling with UML,” in *Membrane Transporters in Drug Discovery and Development*, vol. 637, Q. Yan, Ed. Totowa, NJ: Humana Press, 2010, pp. 23–45. doi: 10.1007/978-1-60761-700-6_2.

- [30] E. Bornberg-Bauer, "Conceptual data modelling for bioinformatics," *Briefings in Bioinformatics*, vol. 3, no. 2, pp. 166–180, Jan. 2002, doi: 10.1093/bib/3.2.166.
- [31] E. Cerami et al., "The cBio Cancer Genomics Portal: An Open Platform for Exploring Multidimensional Cancer Genomics Data," *Cancer Discovery*, vol. 2, no. 5, pp. 401–404, May 2012, doi: 10.1158/2159-8290.CD-12-0095.
- [32] The Galaxy Community et al., "Galaxy for accessible, reproducible, and collaborative data analyses: 2026 update," *Nucleic Acids Research*, vol. 54, no. W1, pp. W105–W116, Jul. 2026, doi: 10.1093/nar/gkag469.
- [33] G. Fischer, "End User Development and Meta-Design: Foundations for Cultures of Participation," *Journal of Organizational and End User Computing (JOEUC)*, vol. 22, no. 1, pp. 52–82, Jan. 2010, doi: 10.4018/joeuc.2010101901.
- [34] "Automatic generation of GUIs from conceptual models using an MDE approach," TEL – Theses Online. <https://theses.hal.science/tel-02967486/> (accessed Sep. 29, 2026).
- [35] M. Trellet, N. Férey, J. Flotyński, M. Baaden, and P. Bourdot, "Semantics for an Integrative and Immersive Pipeline Combining Visualization and Analysis of Molecular Data," *Journal of Integrative Bioinformatics*, vol. 15, no. 2, Jun. 2018, doi: 10.1515/jib-2018-0004.
- [36] "Model Driven Architecture in Containers-Based Software Development," Springer Nature Link. https://link.springer.com/chapter/10.1007/978-3-030-32861-0_10 (accessed Sep. 29, 2026).
- [37] J. Köster and S. Rahmann, "Snakemake—a scalable bioinformatics workflow engine," *Bioinformatics*, vol. 28, no. 19, pp. 2520–2522, Oct. 2012, doi: 10.1093/bioinformatics/bts480.
- [38] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature Biotechnology*, vol. 35, no. 4, pp. 316–319, Apr. 2017, doi: 10.1038/nbt.3820.