ChatGPT

Sequence Alignment and Comparison Tool with Antibody-Specific Features

1 Overview

Modern immunology and protein engineering projects often require the alignment and comparison of large numbers of antibody or protein sequences. General purpose alignment programs (e.g., CLUSTAL Ω, MUSCLE, MAFFT) were not designed to handle highly variable B-cell receptor (BCR)/antibody sequences and typically provide no immunoglobulin-specific annotation [1]. Antibodies contain conserved framework regions (FRs) flanking highly variable complementarity-determining regions (CDRs); numbering schemes such as **Kabat**, **IMGT**, **Chothia** and **Martin** define positions and insertion points within FRs and CDRs [2]. Proper numbering and region annotation are critical for accurate alignment, humanization and engineering. The proposed tool will provide a unified service for sequence alignment and comparison with built-in antibody annotation and visualization capabilities.

2 Goals and Scope

- Accept multiple protein sequences via an API (multipart upload or direct text) in FASTA format.
- Align sequences using one or more proven algorithms:
- **Pairwise alignment** using global (Needleman–Wunsch) or local (Smith–Waterman) algorithms, which can align whole sequences or specific regions [3].
- **Multiple sequence alignment (MSA)** using CLUSTAL Ω/MUSCLE/MAFFT and optionally a custom antibody-aware strategy similar to Abalign (alignment guided by numbering schemes) [4].
- Provide antibody-specific annotation and numbering:
- Use ANARCI to annotate variable domains with numbering schemes (IMGT, Kabat, Chothia, Martin, AHo) [5].
- Highlight framework vs CDR positions and allow region extraction.
- Generate comparative analyses and visualizations:
- **Dendrograms/phylogenetic trees:** compute pairwise sequence distances and build hierarchical clustering or maximum-likelihood trees (e.g., via FastTree) [6].
- **Position-Specific Scoring Matrices (PSSMs):** build PSSMs from MSAs to quantify positional conservation; highly conserved positions receive high scores while weakly conserved positions have scores near zero [7] and can be used to evaluate framework mutation frequencies [8].
- **Sequence logos:** visualize conservation across alignments; logos stack residues at each position with heights proportional to their frequencies [9].
- **Consensus sequences:** derive consensus and identify variable positions.
- Provide options to group sequences (e.g., heavy vs light chains, germline families) and generate separate alignments/logos per group.
- Expose results via API endpoints returning data (JSON, Newick) and images (PNG/SVG). In addition to alignment-centric endpoints, provide an **annotation endpoint** that accepts a list of antibody sequences and returns rich annotations—CDR and framework boundaries, predicted chain type/

isotype, nearest germline gene assignment and species attribution (percent identity to human, mouse, other primates such as cynomolgus, etc.).

- Provide a **React/TypeScript** user interface from the outset (not merely optional) for uploading sequences, choosing analysis options and visualising results. Use **D3.js** for dendrograms, sequence logos and PSSM heatmaps, enabling interactive zooming and tooltips.
- Ensure the tool can run on commodity hardware and comfortably handle up to roughly **200 sequences**. Extreme scalability to thousands of sequences is not a design goal for the initial version.

3 Functional Requirements

3.1 Input and Data Handling

1. **FASTA upload:** Users can send a multi-sequence FASTA file via a `POST /upload` endpoint. Each record contains an identifier and an amino-acid sequence. The server validates that sequences are composed of valid amino-acid characters and are at least a minimum length (e.g., 10 residues).
2. **Direct sequences:** An alternative `POST /align` endpoint accepts a JSON payload with an array of identifiers and sequences along with alignment parameters.
3. **Antibody annotation parameters:** Users may specify chain type (heavy, kappa, lambda or unknown), species (human/mouse/other) and preferred numbering scheme (IMGT, Kabat, Chothia, Martin, AHo). These parameters are passed to ANARCI for numbering [10].
4. **Alignment options:** Users choose algorithms (`pairwise`, `global`, `local`, `muscle`, `clustalo`, `mafft`, `antibody_custom`), substitution matrices (BLOSUM62, PAM250) and gap penalties.
5. **Grouping options:** Users may provide group labels for sequences or request automatic grouping by sequence length or germline V/J assignment.

3.2 Sequence Alignment

1. **Pairwise alignment:** For each pair of sequences, compute global or local alignment using Biopython's `pairwise2` module or SciPy's dynamic programming. Provide alignment scores and aligned sequences with gaps.
2. **MSA using external tools:** Use Biopython wrappers around CLUSTAL Ω, MUSCLE or MAFFT. The tool will call the installed executable (e.g., `clustalo -i input.fa -o output.fa --outfmt=clu`) and parse the result. If none are installed, fall back to a built-in progressive alignment implementation.
3. **Antibody custom alignment:** When antibody annotation is requested, first number each sequence with ANARCI. Align sequences against a consensus template for the chosen scheme, inserting gaps at predefined positions in FRs and CDRs as described for Abalign [11]. This ensures that CDRs align properly even when their lengths differ. Provide an option to treat CDRs separately (e.g., align CDR1, CDR2, CDR3 independently).

3.3 Annotation and Numbering

1. **ANARCI integration:** Use the Python `anarci` package to number antibody sequences. ANARCI aligns a given sequence to a library of hidden Markov models representing germline V gene families and annotates residues with the chosen scheme [10].
2. **Framework/CDR extraction:** Once numbered, label FR1–FR4 and CDR1–CDR3 positions using published definitions [12]. Store annotation metadata with the sequence.

3. **Germline assignment:** Report the predicted germline V gene and species from ANARCI along with the sequence.

4. **Annotation endpoint service:** Provide a `POST /annotate` endpoint (see Section 3.5) that accepts a list of sequences and returns, for each sequence, the numbered positions, FR/CDR boundaries, chain type/isotype prediction, nearest germline gene, percent identity to human/mouse/other primate germlines and species inference. This endpoint leverages ANARCI for numbering, a curated germline database for nearest germline and species scoring, and simple heuristic rules (e.g., constant region motifs) for isotype prediction.

3.4 Visualization and Analysis

1. **Dendrogram/phylogenetic tree:**
2. Compute pairwise distance matrix using percent identity or substitution score per site. Use `scipy.cluster.hierarchy.linkage` to build UPGMA or neighbor-joining dendrogram, or call FastTree 2 when installed to construct a maximum-likelihood tree (FastTree uses heuristics such as nearest-neighbor interchanges and subtree-pruning-regrafting to infer phylogenies and is 100–1,000× faster than other methods [6]).
3. Return the tree in Newick format. Generate a dendrogram plot (PNG/SVG) with branch lengths and labels. Provide options to color leaves by germline family or group.
4. **PSSM generation:**
5. From an MSA, count amino-acid occurrences at each position. Convert counts to position-specific probability matrix; then compute log-odds scores relative to background frequencies (e.g., uniform or BLOSUM frequencies). A PSSM specifies scores for observing particular residues at specific positions [13] and highlights highly conserved positions [14].
6. Provide PSSM in tabular JSON (positions × 20 amino acids) and export as CSV. Offer to visualize it as a heatmap.
7. **Sequence logos:**
8. Use the `logomaker` or `weblogo` Python package to create sequence logos. Each position displays stacked letters whose heights reflect residue frequency; logos highlight conserved regions [9].
9. Support separate logos for FRs and CDRs or for user-defined groups. Save logos as PNG/SVG.
10. **Consensus and variability:** Compute consensus sequence and report per-position identity and entropy. Highlight positions with high variability. Provide summary statistics (mean identity, CDR length distribution, etc.).

3.5 API Endpoints

Method & endpoint	Description	Request parameters	Response
`POST /upload`	Upload one or more sequences in FASTA format.	Multipart form field `file` (FASTA) or `text` (raw sequences). Optional `group`, `chain_type`, `numbering_scheme`.	JSON with dataset ID, number of sequences, warnings.

Method & endpoint	Description	Request parameters	Response
`POST /align`	Align sequences in uploaded dataset or provided inline.	JSON containing `dataset_id` or `sequences` array, `method`, `matrix`, `gap_open`, `gap_extend`, `numbering_scheme`, `grouping` parameters.	JSON with alignment ID.
`GET /alignment/{id}`	Retrieve MSA and aligned sequences.	Query `format` (clustal, fasta) and `annotate` (true/false).	JSON or plain text alignment.
`GET /tree/{id}`	Retrieve dendrogram.	Query `format` (newick, svg, png), `color_by` (group, germline)	File or JSON.
`GET /pssm/{id}`	Retrieve PSSM for alignment.	Query `format` (json, csv), `normalize` (true/false)	JSON/CSV.
`GET /logo/{id}`	Retrieve sequence logo image.	Query `region` (all, FRs, CDRs), `groups`, `format` (png, svg).	Image file.
`DELETE /dataset/{id}`	Delete stored dataset and associated analyses.	None	Confirmation message.
`POST /annotate`	Annotate one or more antibody sequences.	JSON with `sequences` (list of `{id, sequence}`), optional `numbering_scheme`, `species_hint`.	JSON containing per-sequence annotations: FR/CDR boundaries, numbering, predicted chain type/isotype, nearest germline gene, percent identity to human/mouse/other germlines and species inference.

3.6 User Interface

In addition to the API, the tool **must** provide a full-featured web interface built with **React** and **TypeScript**. The front-end will enable users to:

- Upload FASTA files or paste sequences.
- Select alignment algorithms, numbering schemes and visualisation options.
- Trigger the annotation endpoint and view FR/CDR boundaries, germline assignments, isotype predictions and species attribution for each sequence.

- Display multiple sequence alignments, dendrograms, PSSM heatmaps and sequence logos interactively. Visualisations will be implemented with **D3.js** to support zooming, panning and tooltips.

The front-end communicates with the API via REST calls and handles asynchronous job statuses. It packages the analysis results into a cohesive user experience suited for datasets of $\leq$200 sequences.

4 Non-Functional Requirements

- **Performance:** The tool is optimised for datasets up to roughly **200 sequences** on a server with ~4 GB RAM. Long-running tasks (MSA, tree building) will still be executed asynchronously and cached, but supporting tens of thousands of sequences is out of scope for the initial version.
- **Scalability:** The architecture should allow horizontal scaling (multiple worker processes). Tree computation and logo generation can run in background jobs.
- **Modularity:** Separation of concerns between API layer, sequence processing, annotation and visualization. Each component should be testable independently.
- **Extensibility:** Future support for nucleotide sequences, additional numbering schemes or machine-learning-based antibody design can be added.
- **Security:** Validate and sanitize inputs. Do not execute arbitrary command-line input. Limit file size and enforce timeouts on external tools. Provide CORS configuration for API.

5 System Architecture

5.1 Components

1. **API server:** Implemented with FastAPI (Python) for asynchronous endpoints and automatic OpenAPI documentation. Handles request parsing, authentication (optional API keys) and dispatch to processing services.
2. **Sequence processing module:** Responsible for reading FASTA/JSON input, validating amino-acid sequences, grouping sequences and computing pairwise distances. Utilizes Biopython for file parsing and pairwise alignment.
3. **Alignment engine:** Encapsulates calls to external alignment programs (CLUSTAL Ω, MUSCLE, MAFFT). Uses Python `subprocess` with timeouts. Implements an in-memory progressive alignment fallback for small numbers of sequences.
4. **Antibody annotation engine:** Uses the `anarci` library to number sequences with the selected scheme and annotate FR/CDR boundaries. Stores mapping between raw sequence indices and numbered positions, predicts chain type/isotype based on sequence motifs, determines the closest germline V and J genes using sequence identity and a germline database, and calculates species attribution (percent identity to human, mouse, cynomolgus, etc.). Exposes these capabilities through the `POST /annotate` API.
5. **Phylogenetic tree builder:** Computes distance matrices and constructs dendrograms using SciPy's `linkage` or executes FastTree when installed. Converts trees to Newick strings and renders them using `matplotlib` or `ete3`.
6. **PSSM generator:** Implements functions to compute position-specific probability matrices and log-odds PSSMs [15]. Supports weighting schemes and pseudocounts.
7. **Logo generator:** Wraps `logomaker`/`weblogo` to produce sequence logos; ensures CDRs and FRs can be drawn separately.

8. **Storage/cache layer:** Stores uploaded sequences, alignment results, PSSMs and images (e.g., in a local file system or SQLite). Associates each result with a unique ID returned to the user.
9. **Front-end (React/D3):** A single-page application built with **React** and **TypeScript** that communicates with the API via HTTP. It uses **D3.js** for interactive visualisations (dendrograms, sequence logos, PSSM heatmaps) and manages file uploads, parameter selection and display of annotation results. It bundles and serves static assets via the backend or a separate static server.

5.2 Data Flow

1. User uploads sequences via API. The API server saves raw sequences and returns a dataset ID.
2. User requests an alignment. The server retrieves sequences, passes them to the alignment engine with selected options and obtains an MSA. If antibody annotation is enabled, sequences are numbered before alignment and alignment is performed on the numbered positions; the resulting MSA is then mapped back to raw sequences.
3. Annotation engine annotates each sequence with FR/CDR boundaries and germline assignments. Annotation information is stored alongside alignment.
4. Phylogenetic tree builder computes a tree from the MSA. PSSM generator calculates conservation metrics and PSSMs. Logo generator produces sequence logos. Each product is stored and served via its own endpoint.

6 Algorithmic Details and Justification

6.1 Sequence Alignment

- **Pairwise alignment:** Use dynamic programming to compute global (Needleman–Wunsch) or local (Smith–Waterman) alignments. Pairwise alignment identifies conserved regions and is suitable for comparing two sequences or building distance matrices [3] .
- **Multiple sequence alignment:** Progressive alignment methods (e.g., CLUSTAL Ω, MUSCLE) are widely used. They align the most similar sequences first and progressively add sequences based on a guide tree. For antibody sequences, general MSA methods can misalign CDRs because they insert gaps at arbitrary positions. Abalign addresses this by inserting gaps only at numbered positions and pre-annotated insertion points [16] . Our antibody custom alignment will follow this concept: number sequences using ANARCI, align them to a consensus template (including FR/CDR boundaries) and insert gaps at predefined positions in FRs and CDRs.
- **Antibody numbering:** ANARCI aligns sequences to hidden Markov models representing germline gene families, predicts domain type and species, and annotates residues with numbering schemes such as IMGT, Kabat, Chothia, Martin or AHo [10] . Numbering ensures residues from different sequences correspond to equivalent structural positions.

6.2 PSSM and Conservation

A position-specific scoring matrix (PSSM) specifies the score for observing a particular residue at a given position and is derived from an alignment [13] . To compute a PSSM:

1. For each alignment column, count occurrences of each amino acid; convert to probabilities `P_i` by dividing counts by the number of sequences [17] .

2. Compute log-odds scores: `score(i, pos) = log2(P_i / q_i)`, where `q_i` is the background frequency (e.g., uniform 1/20 or derived from a large protein database). Highly conserved positions yield high scores; weakly conserved positions score near zero [14].
3. Store the PSSM in JSON/CSV. Provide optional pseudocounts to avoid negative infinity scores [18].

PSSMs built from large human antibody repertoires show that framework mutation frequencies are consistent across individuals and correlate with germline families [8]. Using PSSMs in antibody engineering can help identify mutations that improve developability [19].

6.3 Sequence Logos

A sequence logo stacks residues at each alignment position with heights proportional to their frequencies and the overall stack height proportional to information content [9]. Sequence logos highlight conserved motifs and variable regions and are particularly useful for visualizing CDR diversity. Grouped logos (e.g., heavy vs light chains or different germline families) can reveal functional differences [20].

6.4 Phylogenetic Trees

Phylogenetic trees model evolutionary relationships. FastTree 2 constructs approximately maximum-likelihood trees and adds heuristics (subtree pruning-regrafting and nearest-neighbor interchanges) to improve accuracy without sacrificing scalability; for large alignments it is 100–1,000× faster than traditional maximum-likelihood methods [6]. If FastTree is not available, a UPGMA or neighbor-joining dendrogram can be built using pairwise distances.

7 Implementation Plan

1. **Environment setup:** Use Python 3.11 with a virtual environment for the backend and Node.js >= 18 with npm/yarn for the frontend. For Python install `biopython`, `fastapi`, `uvicorn`, `scipy`, `matplotlib`, `logomaker`, `pandas`, `numpy`, `anarci`, `ete3`, and any required DB driver. Install external tools (`clustalo`, `muscle`, `fasttree`) via conda or system package manager when available. For the frontend install `react`, `typescript`, `vite`/`create-react-app`, `d3`, and state-management libraries (e.g., Redux or Zustand).
2. **Project structure:**
3. `app/main.py` – FastAPI application.
4. `app/alignment.py` – functions for pairwise and multiple alignment, including external program wrappers and antibody custom alignment.
5. `app/annotation.py` – functions wrapping `anarci` to number sequences, infer chain type/isotype, assign germline genes and compute species attribution.
6. `app/pssm.py` – functions to compute PSSMs and conservation metrics.
7. `app/logo.py` – functions to generate sequence logos using logomaker.
8. `app/tree.py` – functions to compute distance matrices and trees.
9. `app/storage.py` – simple storage (SQLite or file-based) for datasets and result objects.
10. `app/utils.py` – helpers for FASTA parsing, data validation and error handling.
11. `client/` – React/TypeScript front-end project built using Vite or Create React App. Contains `src/components/` for forms, tables and visualisations, `src/hooks/` for API interaction, `src/pages/` for different views and D3 visualisations (dendrogram, logo, PSSM heatmap), and build scripts.

12. **API implementation:** Define endpoints as described in Section 3.5. Use Pydantic models for request/response schemas and include documentation via automatic OpenAPI.
13. **Asynchronous tasks:** For long-running operations (MSA, tree building) use `BackgroundTasks` from FastAPI or integrate Celery/RQ for job management. Return a job ID and allow clients to poll for completion.
14. **Testing:** Write unit tests for each module (alignment, annotation, PSSM, logo, tree). Use synthetic sequences and known alignments to validate correctness. Provide integration tests for API endpoints.
15. **Deployment:** Package the backend and frontend into a multi-stage Docker image. The first stage builds the React frontend (`npm install` and `npm run build`). The second stage installs Python dependencies, system tools and copies the compiled frontend into a directory served by the backend (or served by a lightweight static server such as Nginx). Expose API on port 8000 and serve the React app on the same origin or via a proxy. Provide a `Dockerfile` and `docker-compose.yml` to build and run the service.
16. **Documentation:** Generate API documentation (OpenAPI/Swagger) automatically. Provide usage examples, parameter descriptions and examples of output for each endpoint. Document limitations (e.g., performance degradation for >1000 sequences, supported species).
17. **Security & maintenance:** Validate sequences and file formats. Limit file sizes (e.g., 10 MB) and number of sequences per request (e.g., 5 000). Use sanitized filenames and restrict external program arguments. Provide logging and error reporting.

8 Future Improvements

- **Enhanced interactive features:** Expand visualisation capabilities beyond the core React/D3 implementation. For example, incorporate 3-dimensional structure viewers or integrate with antibody modelling tools to visualise CDR loops. Support drag-and-drop file uploads and real-time editing of alignments.
- **Machine-learning integration:** Incorporate models that predict developability, stability or binding affinity using features derived from PSSMs and antibody numbering.
- **Batch processing & job queues:** Integrate with a message queue (e.g., RabbitMQ) to handle large datasets and run tasks on worker nodes.
- **AIRR compatibility:** Support reading and writing of AIRR-formatted repertoires and integration with VDJtools, MiXCR and other repertoire analysis pipelines [21].
- **Extended numbering schemes:** Add support for additional schemes (Honneger/AHo, GCT) and numbering of nanobody (VHH) sequences.
- **Nucleotide support:** Extend the tool to handle nucleotide sequences, translate them in-frame and annotate V(D)J recombination events.

9 Citations

- **Antibody MSA limitations & antibody-aware alignment:** Abalign, a BCR-specific MSA tool, demonstrates that traditional MSA methods (CLUSTAL Ω, MUSCLE, MAFFT) struggle with massive BCR data and lack immunoglobulin-specific information. Abalign numbers sequences using schemes such as IMGT, Kabat, Chothia and Martin before aligning, resulting in improved speed and accuracy [1] [4].
- **Numbering schemes:** Differences between Kabat, IMGT, Chothia and Martin numbering schemes lie in insertion points for FR and CDR regions; Kabat is sequence-based and less flexible, IMGT uses

germline alignment, Chothia is structure-based, and Martin updates Chothia to account for more variability [12]. Numbering standardizes variable regions and is critical for antibody engineering.

- **ANARCI tool:** ANARCI numbers antibody sequences by aligning them to hidden Markov models of germline V genes and supports multiple numbering schemes (IMGT, Chothia, Kabat, Martin, AHo); it predicts domain type and species [10].
- **Sequence alignment basics:** Pairwise alignment can be global (Needleman–Wunsch) or local (Smith–Waterman) [3]. MSAs are typically built via progressive algorithms that rely on guide trees [22].
- **Sequence logos:** Logos display stacked letters at each position; the height reflects residue conservation and are widely used to visualize motifs [9].
- **PSSMs:** A PSSM specifies scores for observing particular residues at specific positions and is constructed by counting residues in an alignment, computing probabilities and log-odds scores [15]. In PSI-BLAST, PSSMs capture conservation patterns in an alignment and assign higher scores to conserved positions [7].
- **Phylogenetic trees:** FastTree 2 builds approximately maximum-likelihood trees using heuristics (NNIs, SPRs) to achieve high accuracy with much lower computational cost than full ML methods [6].

---

[1] [4] [11] [16] [21] [22] Abalign: a comprehensive multiple sequence alignment platform for B-cell receptor immune repertoires - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC10320167/

[2] [12] Antibody numbering schemes and CDR definitions
https://pipebio.com/blog/antibody-numbering

[3] Understanding Sequence Alignment - Geneious
https://www.geneious.com/guides/understanding-sequence-alignment

[5] [10] SAbPred: ANARCI
https://opig.stats.ox.ac.uk/webapps/sabdab-sabpred/sabpred/anarci/

[6] FastTree 2 – Approximately Maximum-Likelihood Trees for Large Alignments - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC2835736/

[7] [14] PSI-BLAST Tutorial - Comparative Genomics - NCBI Bookshelf
https://www.ncbi.nlm.nih.gov/books/NBK2590/

[8] [19] Regulatory Approved Monoclonal Antibodies Contain Framework Mutations Predicted From Human Antibody Repertoires - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC8503325/

[9] [20] MetaLogo: a heterogeneity-aware sequence logo generator and aligner - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC8921662/

[13] [15] [17] [18] Position-specific score matrices - Species and Gene Evolution
https://cs.rice.edu/~ogilvie/comp571/pssm/