



### Tech Stack

#### 1) Frontend (User Interface)

- **Languages:** HTML, CSS, JavaScript
- **Frameworks/Libraries:**
  - **Bootstrap / TailwindCSS** → responsive UI design
  - **D3.js / Chart.js** → data visualization (graphs, metrics, analytics)
  - **3Dmol.js / NGL.js** → 3D visualization of protein structures, docking, and mutations
- **Role:** Provides a secure portal for users to upload genomic/protein/clinical data and interact with results through charts, dashboards, and 3D molecular visualizations.

#### 2) Backend (Core Engine)

- **Language:** Python (primary)
- **Frameworks:** Flask / FastAPI (REST APIs, secure data handling)
- **Machine Learning & Bioinformatics Libraries:**
  - **scikit-learn** → classification, clustering, regression for data preprocessing and mutation pattern recognition
  - **PyTorch / TensorFlow** → deep learning for sequence modeling (mutation forecasting, drug design)
  - **RDKit** → cheminformatics (SMILES parsing, drug modification, molecular similarity searches)
  - **BioPython** → sequence alignment, genome parsing, protein structure handling
- **Algorithms & Models Used:**
  - **Mutation Prediction:** Hidden Markov Models (HMM), Markov Chain Monte Carlo (MCMC), LSTM/GRU Neural Networks
  - **Protein Folding & Docking:** Molecular Docking (AutoDock Vina), binding energy calculations ( $\Delta G$  approximation), molecular dynamics (OpenMM/GROMACS optional)
  - **Drug/Antidote Optimization:** Generative AI models (VAE/GANs) for new molecule generation, SAR/QSAR analysis for refining drug candidates
  - **Clinical Simulation:** PBPK/PKPD modeling for drug metabolism, dosage prediction, and toxicity profiling

#### 3) Database & Storage

- **Database:** PostgreSQL → structured data (user data, metadata, results)
- **Cloud Storage:** AWS S3 → large biological datasets (genomes, protein structures, docking results, drug libraries)

- **Knowledge Graph (optional):** Neo4j → mapping relationships between mutations, symptoms, and drug candidates
- **Cache Layer:** Redis → for fast access to commonly queried results

## 5) Integration & Workflow Flow

- **User** → Frontend Portal: Uploads viral genomes, protein structures, or clinical datasets.
- **Backend (Python API):** Validates and routes the data to storage.
- **AWS Cloud Storage:** Stores raw data; metadata saved in PostgreSQL.
- **Processing Pipeline (ML/AI):**
  - Mutation prediction (HMM, MCMC, LSTM)
  - Protein docking & binding affinity calculation (RDKit + AutoDock)
  - Antidote optimization (generative AI + QSAR models)
  - In-silico clinical simulation (PBPK/PKPD models)
- **Results Storage:** Predictions and metrics stored back in PostgreSQL + S3.
- **Visualization Layer:** Results delivered via APIs to frontend → displayed as graphs, reports, and interactive 3D models.
- **Feedback Loop:** Validated results/research added back into the knowledge base for continuous model improvement.

## 6) Development Environment

- **IDE:** Visual Studio Code
- **Version Control:** Git + GitHub
- **Containerization (Optional):** Docker (to package ML models and backend services for scalable deployment)