GeneVision – Predictive Genetics & Disease Impact Simulator

□ Problem Statement

Lack of accessible tools for exploring genetic modifications:

Scientists, students, and researchers often struggle to visualize how changes in DNA can affect an organism's appearance or biological functions without advanced simulations or lab experiments.

Difficulty in understanding disease mechanisms:

Many people, even professionals, find it hard to grasp how viruses or diseases target specific genes, organs, and biochemical pathways, making it challenging to study or explain disease progression and treatment options.

Ethical and experimental barriers in research:

Genetic experiments, such as attempts to recreate extinct species or study disease effects, are expensive, time-consuming, and require specialized labs and equipment, limiting learning and exploration.

□ Our Solution

GeneVision will provide an interactive platform where users can:

Predict and visualize how genetic modifications may alter an organism's physical traits (color, size, patterns) using known gene-trait relationships or AI models.

Explore how gene editing could potentially revive extinct traits or species by simulating the insertion of ancient genes into related living organisms.

Simulate how viruses or diseases affect specific human genes, organs, and biochemical pathways, helping users understand symptoms, susceptibility, and disease progression.

Offer a simplified yet scientifically backed visualization tool that supports education, research planning, and hypothesis testing without needing expensive experiments.

□ Services Offered

Genetic Modification Simulator:

Input gene edits and view changes in appearance or behavior of organisms through visual models and descriptions.

PROF

De-Extinction Concept Model:

Explore how extinct species or traits might be recreated using gene editing techniques applied to living organisms.

Disease Impact Explorer:

Understand how diseases or viruses interact with human genes, which organs they affect, and what biochemical processes are triggered during infection.

Educational Interface:

Provide students, educators, and researchers with interactive tools, animations, and explanations to make complex genetics more understandable.

Research Support:

Help researchers form hypotheses, explore possible gene edits, and communicate effects with visuals before conducting lab experiments. 1. The Genetic Modification Simulator (Core Engine)

This is the heart of the project. It operates in two distinct modes to satisfy both casual users (who want to see cool results) and serious researchers (who need granular control).

- Mode A: The AI Architect (Prompt-to-Simulation) (Your "Service 1" Idea)
  - Workflow: User types a natural language prompt (e.g., "Create a mouse with 5 legs").
  - Mechanism: Our NLP & RAG system scans genetic databases to identify the necessary mutations (e.g., Hox gene duplication).
  - Output: The system generates a 3D model of the mutated organism along with a "Gene Recipe" listing the specific required edits.
- Mode B: The Interactive Workbench (Manual Lab) (Your "Service 2" Idea)
  - Workflow: User loads a standard organism model (e.g., Mouse, Fruit Fly). A sidebar lists all modifiable genes.
  - Mechanism: User manually toggles genes on/off or adjusts expression levels using sliders.
  - Output: The 3D model updates in real-time. If you delete a pigment gene, the mouse turns white instantly.

2. De-Extinction Concept Model

- What it is: A comparative genomics tool for evolutionary biology.
- Workflow: Overlay the genome of an extinct species (e.g., Woolly Mammoth) onto a modern relative (e.g., Asian Elephant).
- Output: A visualization of the "Hybrid" organism and a viability score indicating if such a creature could biologically survive.

3. Disease Impact Explorer

- What it is: A visual pathology simulator.
- Workflow: User selects a human organ (e.g., Lungs) and introduces a pathogen (e.g., SARS-CoV-2).
- Mechanism: The system simulates the biochemical cascade—showing the virus binding to receptors and entering cells.
- Output: A clear, visual narrative of disease progression, ideal for patient education.

4. Educational Interface ("GeneVision Academy")

PROF

- What it is: A gamified wrapper around the "Interactive Workbench."
- Target Audience: Schools and Universities.
- Features: Pre-set "Missions" (e.g., "Mission: Use Mendelian genetics to breed a blue-eyed fly"). It simplifies the complex UI into safe, step-by-step learning modules.

5. Research Support ("In Silico Hypothesis Testing")

- What it is: The professional analytical layer.
- Target Audience: Scientists and Biotech firms.
- Features:
  - Off-Target Prediction: Before a scientist performs a real CRISPR edit, our AI predicts unintended side effects (e.g., "Warning: Editing Gene A may cause liver failure").
  - Statistical Validation: Provides probability scores for experimental success, helping researchers "fail fast" in the simulation before wasting money in the lab. Why Mouse?

        - The MGI (Mouse Genome Informatics) database is the "gold standard" of biology. It has thousands of confirmed "Gene ↔ Visible Trait" connections.
        - Rat data is good, but Mouse data is more visual (coat colors, ear shapes, tail lengths), which is exactly what your project needs to show off.
          Here is your "Weekend MVP" Plan to build a working demo of both services.
          Phase 1: Get the Data (The "Brain")
          You don't need AI for the demo. You need a simple "Lookup Table" (database).

1. Where to download:
    - Go to the International Mouse Phenotyping Consortium (IMPC) or MGI.
    - Specific File: Look for the `Genotype_Phenotype_Assertions` report.
    - What you are looking for: A CSV file that looks like this:
      Gene SymbolAllelePhenotype TermPhenotype IDTyrcAlbinism (White coat)MP:000001LepobIncreased body weightMP:000123Hoxd13spdhPolydactyly (Extra toes)MP:000456KitWWhite spottingMP:000789
      Export to Sheets
      Action Item: Create a small `data.json` file with just 5 clear examples (like the ones above) to test your system. Don't try to load the whole genome yet.
      Phase 2: Get the Visuals (The "Body")
      You need a "Digital Twin" of a mouse.
2. The 3D Model:
    - Go to Sketchfab or TurboSquid and search for "Low Poly Mouse" (free).
    - Download it in glTF or OBJ format.
    - Crucial Step: You need a model that has a "Texture Map" (skin image).
3. The visual tricks (How to cheat for the demo):
    - For Color: You don't need complex 3D morphing. Just swap the Texture Image.
        - Standard: Grey mouse texture.
        - Albino: White mouse texture.
        - Spotted: White/Grey patch texture.
    - For Size (Obesity): Use a standard 3D scaling property in your code.
        - `scale_x = 1.5` (Make it wider).
          Phase 3: Build the Logic (The "Engine")
          We will write a simple Python script (using Streamlit is best for a fast demo) to connect Phase 1 and Phase 2.
          Service 1 Demo: Forward Genetics (Text → Gene)
    - User Input: "I want a fat mouse."
    - Your Code:
        1. Scans input for keywords: "fat", "obese", "heavy".
        2. Looks in `data.json` for "Increased body weight".
        3. Finds Gene: Lep (Leptin).
        4. Output: Displays "Candidate Gene Found: Lep" and triggers the "Wide Mouse" 3D model.
           Service 2 Demo: Reverse Genetics (Gene → Trait)
    - User Input: Selects Tyr (Tyrosinase) from a dropdown menu.
    - Your Code:
        1. Looks in `data.json` for Tyr.

PROF

2. Finds Phenotype: "Albinism".
3. Output: Swaps the mouse texture from "Grey.png" to "White.png".

   Phase 4: How to Present It

   When you show this to your professors or judges, follows this flow:

1. "The Problem": Show a raw Excel sheet of gene data. Say: "This is how scientists see data today. It's boring and hard to understand."
2. "The Solution (Service 2)": Open your tool. Select the Kit gene.
   - Bam! The mouse on screen instantly turns spotted.
   - Say: "We visualize the data instantly."
3. "The Future (Service 1)": Type "Make it a monster." (Pre-program this to make the mouse green or giant).
   - Say: "Our AI will eventually let you design organisms using just words." he Genetic Data (The "Brain")

     You need a file that says: "Gene X = Trait Y". The best source is the Mouse Genome Informatics (MGI) database.
     - Source: MGI Data and Statistical Reports
     - The Specific File You Want: `MGI_Geno_Disease_DO.rpt` OR `MGI_PhenoGenoMP.rpt`.
       - Why: These are plain text files that list Genotypes alongside their Phenotypes (Physical Traits). RAG (Retrieval-Augmented Generation) system is the only scalable way to handle Forward Genetics (Service 1).

         Here is why RAG is necessary instead of just a simple database lookup:
         - Users don't speak "Gene": A user will say "I want a mouse that doesn't get cancer," or "I want a super-strong mouse."
         - Databases speak "Gene": The database only knows Trp53 or Mstn.
         - The RAG's Job: It acts as the translator. It understands that "strong" semantically relates to "muscle mass" and retrieves Mstn (Myostatin) without the user ever knowing the technical term.

           The Architecture: How Your RAG Will Work

           Here is the exact flow you need to build for the "Fat Mouse" example.

           Step 1: User Prompt (The Input)
         - User says: "I want to create a fat mouse."

           Step 2: The Retriever (The Search)
         - What happens: The system doesn't just look for the word "fat." It converts the user's prompt into a Vector Embedding (a mathematical representation of meaning).
         - It searches for concepts like: "Obesity," "Adipose tissue," "Body weight increase," "Metabolic syndrome."
         - It retrieves from your DB:
           - Gene: Lep (Leptin) → Phenotype: Severe Obesity.
           - Gene: Lepr (Leptin Receptor) → Phenotype: Obesity and diabetes.
           - Gene: Cpe (Carboxypeptidase E) → Phenotype: Late-onset obesity.

PROF

Step 3: The Generator (The LLM)

- What happens: The LLM (like Gemini or GPT) takes the user's prompt "Fat Mouse" AND the retrieved data (Lep, Lepr, Cpe).
- It constructs a response:
  - "To create a 'fat mouse,' the most effective method is a knockout of the Lep (Leptin) gene. This disrupts appetite regulation, leading to rapid weight gain."

Step 4: The Visual Prompt Generator (The Bridge)

- Crucial Step: The LLM also generates a technical prompt for your image/3D generator.
  - Output for Image Gen: "A laboratory mouse with the ob/ob phenotype, exhibiting severe obesity, rounded body shape, excess adipose tissue, white coat, photorealistic, 8k."
  - Output for 3D Gen: `set_morph_target("body_width", 1.8)` or `apply_texture("obese_skin")`.

Why this impresses judges:

1. Semantic Understanding: If you type "A mouse that can't stop eating," a simple SQL query for "fat" would fail. RAG will succeed because it understands the meaning of the phrase matches the function of Leptin.

2. Handling Complexity: If a user asks for "A mouse with red eyes and big muscles," the RAG can retrieve two different genes (Tyr for eyes, Mstn for muscles) and combine them.
   Tech Stack Update for RAG
   Since we are agreeing on RAG, here is what you need to add to your stack:

3. Vector Database: ChromaDB or FAISS (Both are free, open-source, and run locally on your laptop).

4. Embedding Model: HuggingFace (`all-MiniLM-L6-v2`) to turn text into vectors.

   1. Orchestrator: LangChain (to glue the prompt, the search, and the answer together).

PROF

4. The Two-Mode Strategy (AI Architect + Interactive Workbench)

Brilliant design choice. This serves both casual users AND researchers without compromise.
Mode A (prompt-to-simulation) is your "wow factor" for demos.
Mode B (manual gene toggling) is your credibility builder for scientists.

2. Starting with Mouse Models

Strategically correct. MGI is the most complete phenotype database in existence.
The phenotype-genotype mappings are well-documented, giving you real scientific backing.
Visual traits (coat color, body size) are perfect for a proof-of-concept demo.

3. The RAG Architecture for Service 1

Absolutely necessary. You correctly identified that users don't speak "gene language."
Your semantic search approach (user says "fat" → system retrieves "Lep/Lepr/obesity pathways") is the ONLY way to make this accessible.
The flow: User Prompt → Vector Retrieval → LLM Context Assembly → Visual Generation is architecturally correct.

4. The Weekend MVP Plan

Pragmatic and executable. Starting with 5 hardcoded gene-phenotype pairs is the right way to validate the concept before scaling.
Using texture swaps instead of complex morphing for the demo is smart corner-cutting.

⚠ Critical Gaps & Necessary Corrections
Gap 1: Oversimplification of Gene-Phenotype Relationships
Your Assumption:

"Gene X = Trait Y" is a clean lookup table.

Reality:

Polygenic Traits: Most visible traits result from multiple genes interacting.

Example: Coat color isn't just Tyr. It's Tyr + Kit + Mc1r + environmental factors.
Your system needs to handle combinatorial genetics, not just single-gene lookups.

What This Means for Your MVP:

For the demo: Stick to monogenic traits (single gene = clear outcome). Examples:

Tyr knockout → Albinism □
Lep knockout → Obesity □
Pax6 mutation → No eyes □

PROF

For the production version: You'll need a Bayesian network or phenotype scoring system to weight multi-gene contributions.

Gap 2: The 3D Visualization Challenge
Your Plan:

Use texture swaps and scaling for traits.

What's Missing:
You haven't addressed morphological changes that can't be solved with textures:

Extra limbs (polydactyly from Hoxd13)
Missing organs (anophthalmia from Pax6)
Skeletal deformities

Corrected Approach:

For MVP: Limit to traits that CAN be shown via textures/scaling:

Colors □
Size/weight □
Patterns □

For Future: You'll need parametric 3D morphing (tools like Blender's shape keys or procedural generation). This is a 6-month research project, not a weekend hack.

Critical Recommendation:
State clearly in your pitch: "Version 1.0 focuses on pigmentation and size traits. Structural morphology is our Phase 2 roadmap." This shows honesty and planning.

Gap 3: The Disease Impact Explorer Is Underspecified
Your Vision:

Simulate how viruses affect genes/organs/biochemical pathways.

What's Unclear:

Are you simulating genetic susceptibility (e.g., ACE2 expression levels affecting COVID severity)?
Or real-time infection dynamics (viral replication rates, immune response)?

These are VERY different problems:

Option A (Genetic Susceptibility): Feasible. You'd show: "Patient with SNP rs123 in ACE2 has 2.3x higher viral binding affinity."
Option B (Dynamic Simulation): This requires systems biology modeling (differential equations, agent-based models). Not doable in your timeline.

Corrected Scope for MVP:
Focus on static genetic vulnerability mapping:

User selects: "COVID-19"
System shows:

PROF

Affected genes (ACE2, TMPRSS2)
Affected organs (Lungs, Heart)
Risk factors based on known SNPs

Save dynamic simulations for later.

Gap 4: RAG Implementation Complexity
Your Plan:

Use ChromaDB + HuggingFace embeddings + LangChain.

What You're Underestimating:

Data Preparation Hell: MGI data isn't clean. You'll need to:

Parse inconsistent formatting
Resolve gene name synonyms (Lep vs LEP vs ob)
Handle outdated phenotype terms

Prompt Engineering for Multi-Gene Queries:

User: "Make a mouse resistant to cancer and obesity."
Your RAG needs to retrieve both Trp53 (tumor suppressor) and Lep (metabolism), then intelligently combine them in the response.

Corrected Approach:

For MVP: Only handle single-trait queries. Example: "Fat mouse" → Returns one primary gene (Lep).
For Production: Build a query decomposition layer that breaks complex prompts into sub-queries.

□ What's MISSING from Your Architecture
Missing Component 1: Off-Target Effect Prediction
You mentioned this in "Research Support," but didn't specify how you'd do it.
The Reality:

CRISPR edits can cause unintended mutations.
Your system should warn: "Editing Lep may disrupt nearby gene Leprot, causing insulin resistance."

How to Build This (Simplified):

Use genome browser data (UCSC Genome Browser) to check if target genes overlap with regulatory regions.
For MVP: Hardcode known off-target risks for your 5 demo genes.

Missing Component 2: Experimental Feasibility Scoring
Your Problem Statement Says:

"Help researchers plan experiments before expensive lab work."

PROF

What's Missing:
You need to tell researchers: "This edit has a 70% success rate in published studies."
How to Add This:

Scrape PubMed for papers mentioning your target gene + "CRISPR" or "knockout."
Display success statistics: "23 studies attempted Lep knockout. 19 succeeded. 4 reported embryonic lethality."

□ Revised Weekend MVP Checklist
Here's what you should ACTUALLY build for your demo:
Day 1: Data + Backend

Download MGI MGI_PhenoGenoMP.rpt
Extract 5 monogenic traits (Tyr, Lep, Kit, Pax6, Hoxd13)
Create data.json

IM BUILDING RAG SYSTEM FIRST the data is in the data folder