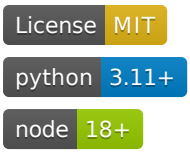
GeneVision

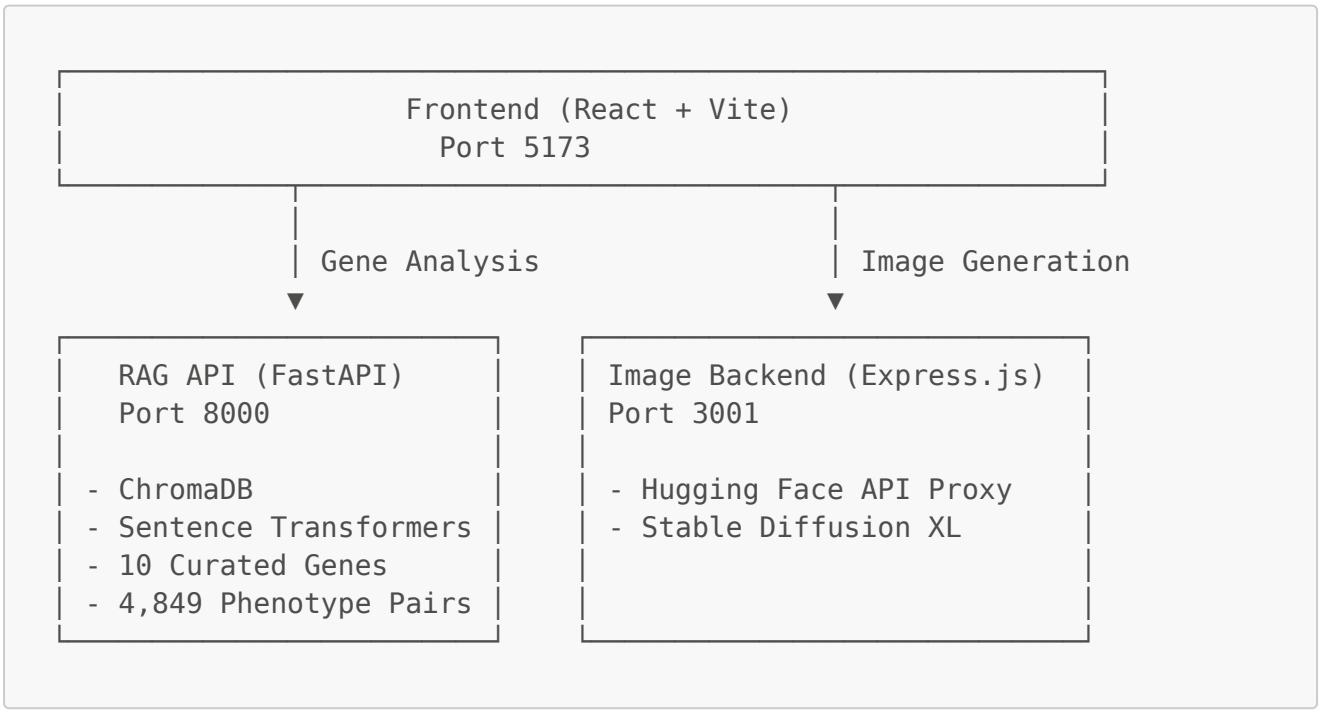
A full-stack genetic modification simulator for mice that combines RAG-based gene analysis with AI-powered visualization. Input natural language queries like and receive scientifically-backed gene recommendations with visual representations.



Features

- **Natural Language Gene Search** - Describe phenotypes in plain English ("3 legs", "white fur", "obese mouse")
- **RAG-Powered Analysis** - Retrieval-Augmented Generation using ChromaDB and sentence transformers
- **Curated Gene Database** - 10 scientifically validated genes from MGI (Mouse Genome Informatics)
- **Confidence Scoring** - High/Medium/Low confidence badges for gene recommendations
- **AI Image Generation** - Visual phenotype representations via Stable Diffusion XL
- **Real-time Chat Interface** - Interactive UI with gene details, alleles, and phenotypes

Architecture



Tech Stack

Backend RAG System:

- FastAPI (Python)

- ChromaDB (Vector Database)
- sentence-transformers/all-MiniLM-L6-v2 (Embeddings)
- MGI Mouse Genome Database

Image Generation:

- Node.js + Express
- Hugging Face Inference API
- Stable Diffusion XL

Frontend:

- React 18
- Vite
- TailwindCSS
- Lucide Icons

Quick Start

Prerequisites

- Python 3.11+
- Node.js 18+
- 8GB+ RAM (for local embeddings)

Installation

1. **Clone the repository**

```
git clone <repository-url>
cd gene-vision
```

PROF

2. **Set up environment variables**

   Create `/home/uday0/code/gene-vision/backend/.env`:

```
HF_API_TOKEN=your_huggingface_token_here
PORT=3001
```

3. **Install dependencies**

   RAG System:

```
cd rag-system
python -m venv venv
```

```
source venv/bin/activate
pip install -r requirements.txt
```

Image Backend:

```
cd backend
npm install
```

Frontend:

```
cd frontend
npm install
```

4. **Build vector database** (first time only)

```
cd rag-system
source venv/bin/activate
python scripts/build_db.py
```

This indexes 4,849 gene-phenotype pairs and takes ~5 minutes.

Running the Application

Option 1: Use the startup script (Recommended)

```
chmod +x start.sh
./start.sh
```

PROF

Option 2: Start services manually

Terminal 1 - RAG API:

```
cd rag-system
source venv/bin/activate
uvicorn src.main:app --host 0.0.0.0 --port 8000
```

Terminal 2 - Image Backend:

```
cd backend
node server.js
```

Terminal 3 - Frontend:

```
cd frontend
npm run dev
```

Stopping the Application

```
./stop.sh
```

Access the Application

- **Frontend**: http://localhost:5173
- **RAG API**: http://localhost:8000
- **API Docs**: http://localhost:8000/docs
- **Image Backend**: http://localhost:3001

Usage Examples

Example Queries

Query	Expected Gene	Confidence	Phenotype
"muscular mouse"	Mstn	High	Increased muscle mass
"pink fur"	Mc1r	Medium	Pink/red coat color
"white fur"	Kit	High	White spotting
"obese mouse"	Lep	Low	Obesity
"3 leg mouse"	Hoxd13	High	Polydactyly/limb defects
"eyeless mouse"	Pax6	High	Eye development defects

PROF

API Usage

Query genes via API:

```
curl -X POST http://localhost:8000/query \
  -H "Content-Type: application/json" \
  -d '{"prompt": "muscular mouse", "top_k": 5}'
```

Response structure:

```
{
  "query": "muscular mouse",
  "genes": [
    {
      "gene_symbol": "Mstn",
      "gene_name": "Myostatin",
      "description": "Negative regulator of muscle growth",
      "aggregate_score": 0.0666,
      "confidence_level": "high",
      "phenotypes": [...],
      "alleles": [...],
      "pubmed_refs": [...]
    }
  ],
  "total_results": 1,
  "warning": null
}
```


Curated Genes

The system returns only these 10 scientifically validated genes:

Gene	Full Name	Phenotype
Tyr	Tyrosinase	Albinism
Mc1r	Melanocortin 1 Receptor	Coat color (red/yellow/brown)
Kit	KIT Proto-Oncogene	White spotting
Lep	Leptin	Obesity
Lepr	Leptin Receptor	Obesity/diabetes
Cpe	Carboxypeptidase E	Late-onset obesity
Pax6	Paired Box 6	Eye development
Hoxd13	Homeobox D13	Polydactyly
Mstn	Myostatin	Muscle growth
Trp53	Tumor Protein 53	Cancer/tumor suppression

PROF

Confidence Scoring

Gene recommendations include confidence levels based on aggregate scores:

- ★★★ **High** (≥0.04): Strong semantic match with query
- ★★ **Medium** (0.02-0.04): Moderate match, relevant but less specific
- ★ **Low** (<0.02): Weak match, may be tangentially related

Note: Genes with scores below 0.01 are automatically filtered out.

Project Structure

```
gene-vision/
├── backend/                    # Image generation service (Express.js)
│   ├── server.js              # Main server with HF API proxy
│   ├── package.json
│   └── .env                   # HF_API_TOKEN
│
├── frontend/                   # React UI (Vite)
│   ├── src/
│   │   ├── App.jsx
│   │   ├── ReconstructionPanel.jsx  # Main chat interface
│   │   └── GeneDetails.jsx          # Gene information panel
│   ├── package.json
│   └── vite.config.js
│
├── rag-system/                 # RAG engine (FastAPI + ChromaDB)
│   ├── src/
│   │   ├── main.py            # FastAPI app
│   │   ├── rag_engine.py      # Core RAG logic
│   │   ├── gene_curator.py    # Curated gene definitions
│   │   └── schemas.py         # Pydantic models
│   ├── scripts/
│   │   └── build_db.py        # Vector DB builder
│   ├── data/
│   │   └── MGI_GenePheno_*.rpt  # MGI phenotype data
│   ├── chromadb_storage/      # Vector database (generated)
│   └── requirements.txt
│
├── start.sh                    # Start all services
├── stop.sh                     # Stop all services
└── README.md                   # This file
```

PROF

How It Works

1. Gene Analysis Pipeline

```
User Query ("muscular mouse")
    ↓
Embedding Generation (all-MiniLM-L6-v2)
    ↓
Vector Search in ChromaDB (4,849 phenotypes)
    ↓
Gene Aggregation & Scoring
    ↓
Filter to Curated Genes (10 genes)
    ↓
```

```
Apply Minimum Score Threshold (≥0.01)
    ↓
Rank by Confidence (High/Medium/Low)
    ↓
Return Top K Results
```

2. Image Generation Pipeline

```
Gene Results + User Query
    ↓
Build Enhanced Prompt
    ↓
Send to Hugging Face API (Stable Diffusion XL)
    ↓
Receive Base64 Image
    ↓
Display in Frontend
```

Known Issues & Limitations

Image Generation Unavailable

Status: Hugging Face API credits depleted

Impact:

- Gene analysis works perfectly
- Image generation shows friendly error message
- System gracefully degrades to "gene analysis only" mode

Solutions:

PROF

1. Add Hugging Face credits ($9/month PRO subscription)
2. Switch to Replicate API (~$0.002/image)
3. Use local Stable Diffusion (requires GPU)

Other Limitations

- Limited to 10 curated genes (by design for educational purposes)
- No support for multi-gene combinations
- Image quality depends on Stable Diffusion prompt engineering

Troubleshooting

RAG API won't start

```
```

```
# Check if port 8000 is in use
lsof -ti:8000

# Rebuild vector database
cd rag-system
python scripts/build_db.py
```

Frontend shows "Connection refused"

```
# Check if backend services are running
curl http://localhost:8000/health
curl http://localhost:3001/health

# Restart all services
./stop.sh && ./start.sh
```

Image generation fails

```
# Check backend logs
tail -f /home/uday0/code/gene-vision/backend/server.log

# Verify HF token is set
cat backend/.env | grep HF_API_TOKEN
```

Development

Adding New Curated Genes

PROF

Edit `rag-system/src/gene_curator.py`:

```
CURATED_GENES = {
    "Foxn1": "Hairless phenotype",  # Add new gene
    # ... existing genes
}
```

Then rebuild the vector database:

```
cd rag-system
python scripts/build_db.py
```

Adjusting Confidence Thresholds

Edit `rag-system/src/rag_engine.py`:

```
def calculate_confidence_level(self, aggregate_score: float) -> str:
    if aggregate_score >= 0.04:  # Adjust these values
        return "high"
    elif aggregate_score >= 0.02:
        return "medium"
    else:
        return "low"
```

Changing Minimum Score Threshold

Edit `rag-system/src/rag_engine.py`:

```
MINIMUM_SCORE_THRESHOLD = 0.01  # Lower = more results
```

API Reference

GET /health

Health check endpoint

Response:

```
{
  "status": "healthy",
  "version": "1.0.0",
  "genes_indexed": 10,
  "phenotypes_indexed": 4849
}
```


POST /query

Query genes by phenotype description

Request Body:

```
{
  "prompt": "muscular mouse",
  "top_k": 5
}
```

Response: See "API Usage" section above

Performance

- **Gene Query Latency:** ~200ms (local embeddings)
- **Image Generation:** 10-30 seconds (HF API)
- **Vector Database Size:** ~50MB
- **Memory Usage:** ~2GB (embeddings loaded in memory)

Contributing

Contributions are welcome! Areas for improvement:

1. Expand curated gene list (add hair length, more colors, etc.)
2. Implement multi-gene query support
3. Add gene interaction warnings
4. Improve image prompt engineering
5. Add test suite
6. Create Docker deployment

License

MIT License - See LICENSE file for details

Acknowledgments

- **MGI (Mouse Genome Informatics)** - Gene-phenotype data
- **Hugging Face** - Model hosting and inference API
- **ChromaDB** - Vector database
- **Sentence Transformers** - Embedding models

Citation

If you use GeneVision in your research, please cite:

PROF

```
@software{genevision2024,
  title={GeneVision: AI-Powered Genetic Modification Simulator},
  author={Your Name},
  year={2024},
  url={https://github.com/yourusername/gene-vision}
}
```

Built with ❤ for genetic research and education

For questions or issues, please open a GitHub issue.