GenTREE

**Team Members:**

M. Leela Prasanthi

R. Prasanna

Ch. Navadeep

K. Vaishnavi

B. Lekhana Sri

**ABSTRACT**

**GenTREE** is an innovative AI-pwered health genealogy platform that helps families citualize track, and analyze health data across generations. It combines family tree visualization with medical intelligence to uncover genetic risk factors, shared health patterns, and preventive care recommendations. By linking ancestry with health analytics, GenTree enables families to understand how hereditary and lifestyle factors influence overall well-being. The platform uses AI-driven insights to predict potential hereditary diseases and offer personalized wellness guidance. It acts as a secure hub for storing and managing health data of multiple family members in one place. GenTREE ensures data privacy and encryption, allowing sharing only with user consent. Its interactive dashboard provides real-time analytics, health trends, and family wellness reports. The system promotes preventive heakthcare and early awareness, helping users take actiion before risk arise. Ultimately, GenTREE empowers familities to make informed, procative, and connected health decisions for a healthier future.

**OVERVIEW**

**GenTREE: Decode your DNA. Discover your Destiny.**

In today’s healthcare landscape, individuals often manage their health in isolation, overlooking critical family-wide patterns. Medical histories are fragmented, forgotten, or inaccessible across generations, making early detection of hereditary conditions difficult. Moreover, there’s no intuitive, secure tool to visualize and analyze these connections.

GenTREE solves this by enabling users to build a digital family health tree, inputting member profiles and medical data to uncover shared conditions and genetic risks. Through AI-driven analysis, the platform offers personalized preventive care suggestions while maintaining strict data privacy and consent-based sharing. By merging genealogy with health insights, GenTREE empowers families to make informed decisions, detect risks early, and embrace proactive wellness together.

**DATA COLLECTING & PREPROCESSING**

*Public Health Datasets:* Supplemented with anonymized datasets from sources like WHO, CDC, and open medical repositories to train the AI model.

*Data Sources User-Submitted Health Profiles:* Family members manually input their medical history, lifestyle habits, and known conditions.

*Synthetic Data Generation:* Used to simulate diverse family health scenarios for testing and model validation. Data Transformation Techniques Data Cleaning Procedures.

*Missing Value Handling:* Null entries (e.g., unknown conditions or ages) were filled using imputation techniques like mean/mode substitution or flagged for user review.

*Duplicate Removal:* Repeated entries across generations were identified and merged using name-matching and relationship heuristics.

*Standardization:* Medical terms were normalized using ICD-10 codes to ensure consistency across user inputs and external datasets.

Extreme values (such as irrational ages or conditions) were identified and either corrected or removed from analysis as part of the outlier detection process.

Techniques for Data Transformation

*Encoding Relationships:* For tree visualisation, family ties (such as parent-child and sibling relationships) were mapped into a graph structure.

*Feature Engineering:* Developed derived features for AI analysis, such as "generational health patterns," "risk score," and "condition clusters."

*Normalisation:* To enhance model performance, scale numerical data (such as age and BMI).

*Tokenisation and Vectorisation:* Using NLP techniques, text-based health descriptions were transformed into numerical vectors for machine learning processing.

How is this different from similarly existing websites?  
  
🔍 1. Family-Centric Health Intelligence

Most health platforms focus on individual records or generic symptom tracking. GenTREE is built around family-wide health mapping, helping users see intergenerational patterns and shared risks — something rarely addressed in mainstream health apps.

🧠 2. AI-Powered Hereditary Risk Detection

Unlike static genealogy tools or basic health trackers, GenTREE uses AI to analyze genetic traits, shared conditions, and lifestyle habits across generations. It doesn’t just store data — it interprets it to offer personalized preventive care suggestions.

🌳 3. Interactive Family Health Tree

Existing genealogy sites (like Ancestry or MyHeritage) focus on lineage, not health. GenTREE combines both, offering a visual health tree that connects medical histories with relationships, making it easy to spot inherited conditions.

🔐 4. Privacy-First Architecture

Many platforms collect data without clear consent flows. GenTREE ensures encrypted storage, consent-based sharing, and user control over who sees what — especially important for sensitive health information.

📊 5. Health Trends Dashboard

GenTREE provides a dashboard view of common conditions, habits, and risks across the family — something most apps don’t visualize. It turns raw data into actionable insights for families.

**Model Architecture**

The Node.js/Express.js backend, which controls the entire analysis, is at the heart of our model architecture. Our backend retrieves the family's health information as JavaScript objects by querying the MongoDB database in response to a request from the frontend. After that, this data is arranged in a structured manner and incorporated into a thoughtfully designed prompt. A pre-trained Large Language Model (LLM) receives this prompt through a secure API call. After analysing the data, the LLM provides plain text insights, which our backend then sends to the user's HTML/JS frontend for display.

**DEPLOYMENT**

*1. Frontend Deployment (HTML, CSS, JS)*

Since our frontend is built with pure HTML, CSS, and JavaScript, we will deploy it as a static site using a service like Netlify, GitHub Pages, or Vercel.

How it Works: We connect our GitHub repository to Netlify. Whenever we push new code to the main branch, Netlify automatically pulls the files and deploys them to its powerful global CDN (Content Delivery Network).

*2.Backend Deployment (Node.js & Express.js)*

Our Node.js and Express.js backend API will be deployed to a PaaS (Platform as a Service) like Heroku or containerized using Docker and hosted on Google Cloud Run.

How it Works (with Heroku): We connect our GitHub repository to Heroku. When we push new code, Heroku detects the package.json file, automatically installs all the necessary Node.js dependencies (npm install), and starts our Express server.

*3. Database Management (MongoDB)*

We will use MongoDB Atlas, a fully managed cloud database service.

How it Works: Instead of installing and managing MongoDB on our own server, we use Atlas to provision a database cluster in the cloud. Our deployed backend application connects to this database cluster using a secure connection string.

*4. Monitoring and Security*

Platform Monitoring: We will use the built-in monitoring tools provided by Heroku and Netlify to watch for errors and track application performance.

Environment Variables: All sensitive credentials, like our MongoDB connection string and any API keys, are stored securely as environment variables. They are never written directly in the code, ensuring our secrets are kept safe and separate from the codebase.

**Instructions for Running the Project**

*1. Prerequisites (Software to Install)*

**Node.js:** This is used to run our backend server. It comes with npm (Node Package Manager), which we'll use to install dependencies. You can download it from [nodejs.org](https://nodejs.org/).

**Git:** This is used to clone the project repository from its source. You can download it from [git-scm.com](https://git-scm.com/).

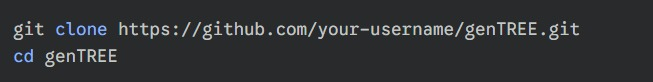
**MongoDB:** You need a running MongoDB database. You have two main options:

* **Recommended:** Sign up for a free **MongoDB Atlas** account ([mongodb.com/cloud/atlas](https://mongodb.com/cloud/atlas)). This is a cloud-based database and is easier to set up.
* **Local Install:** Install **MongoDB Community Server** on your computer.

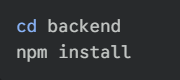
**A Code Editor:** A good editor like **Visual Studio Code** is highly recommended.

*2. Project Setup and Configuration*

1. **Clone the Repository:** Open your terminal or command prompt and run the following command to download the project file



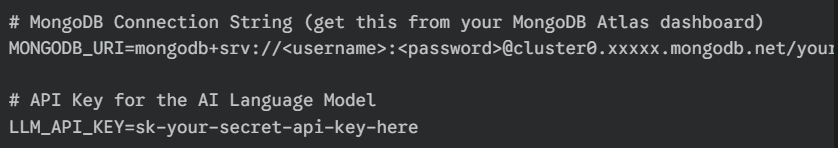
1. **Set Up the Backend:** Navigate into the backend directory and install the necessary Node.js packages.

****

*3.Configure Environment Variables:*

The backend needs to connect to your database and the AI model's API. We store these secrets in an environment file to keep them secure.

* In the backend folder, create a new file named .env
* Add the following content to the .env file, replacing the placeholder values with your actual credentials



*4. Accessing the Project*

Your GenTREE application is now running!

* The **frontend** is accessible in your browser, likely at http://127.0.0.1:8080 or a similar address provided by live-server.
* The **backend API** is running in the background at http://localhost:3000.

**Training Process:**

*How GenTREE's AI Provides Advice:*

Our system doesn't "learn" in the traditional sense; it uses a pre-trained model's vast knowledge and follows a strict set of instructions we provide in real-time.

***Step 1:*** Data Aggregation from the Backend

When a user requests insights, our Node.js backend first pulls the relevant family health data from the MongoDB database. It cleans and organizes this data into a simple, structured format.

* Example of Organized Data:

Self: Age 35, healthy

Father: Age 65, diagnosed with Type 2 Diabetes at 55

Paternal Grandmother: Deceased, had Type 2 Diabetes

Mother: Age 62, has high blood pressure

***Step 2:*** The "Prompt Engineering" - Crafting the Master Instruction

This is the most critical step. Our backend code programmatically builds a detailed set of instructions (a "prompt") that it sends to the AI. This prompt acts as the AI's rulebook for that specific request. It consists of four parts:

* Persona: We tell the AI what its role is.

"You are a helpful and cautious AI assistant for the GenLink platform. Your goal is to provide general wellness information based on family health patterns."

* The Data: We insert the structured data from Step 1.

"Analyze the following family health data: [Data from Step 1 is inserted here]"

* The Specific Task (The User's Question): We tell it exactly what to generate.

"Based on the patterns in this data, provide the following in clear sections:"

* Risk Alert: Identify potential hereditary health risks in simple, non-alarming terms.
* Preventions: Suggest general, evidence-based lifestyle habits to help mitigate these risks.
* Precautions: List common-sense health monitoring steps the user could discuss with a doctor.
* Food Diet: Describe general dietary principles associated with preventing the identified risks.
* The Guardrails (Strict Safety Rules): This is non-negotiable. We give the AI firm commands on what NOT to do.

*IMPORTANT RULES:*

* DO NOT diagnose any disease or condition.
* DO NOT prescribe specific medications or supplements.
* ALWAYS end every response with a clear disclaimer: 'This is not medical advice. Please consult a qualified healthcare professional for personalized guidance.'

***Step 3:*** AI Generation

The backend sends this complete, detailed prompt to the LLM's API. The AI follows all the instructions—the persona, the task, and the safety rules—to generate a coherent, structured response based on its vast pre-existing knowledge about health and wellness.

***Step 4:*** Delivering the Safe Response

The AI sends the text back to our Node.js server. The server does a final check (e.g., ensuring the disclaimer is present) and then sends the formatted information to the user's HTML/JS frontend, where it is displayed in the appropriate sections.

*In Summary:* An Example

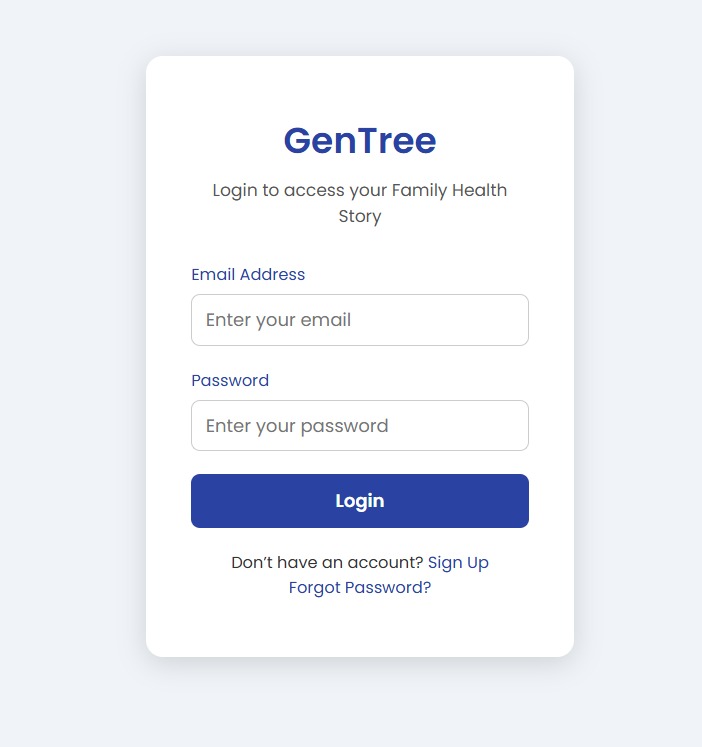
Input Data: History of Type 2 Diabetes.

AI's Generated Output (following our rules):

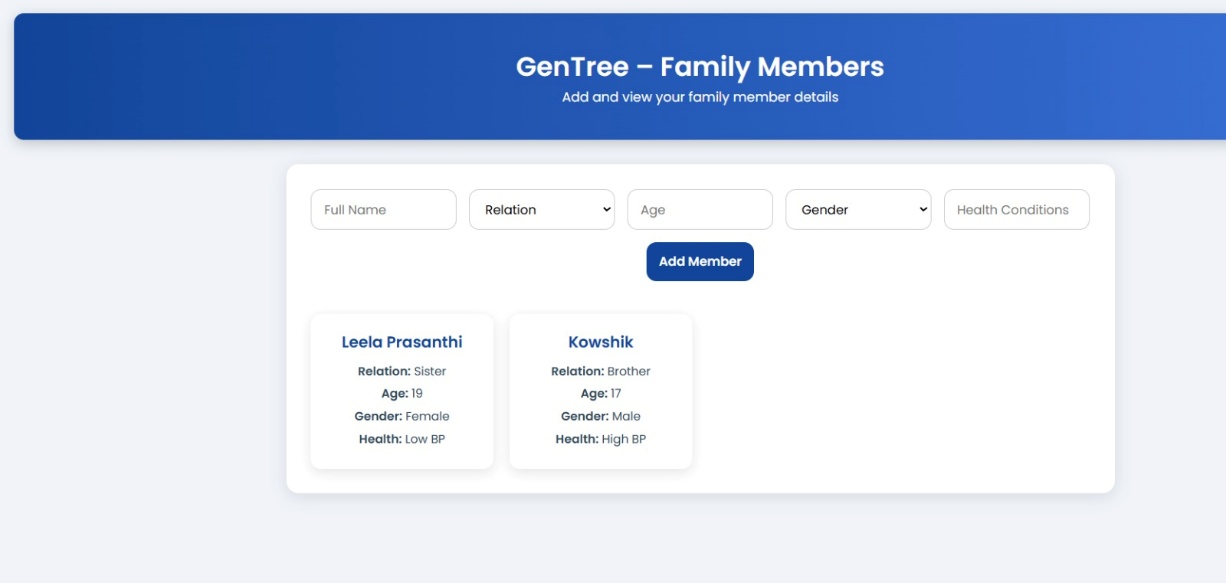
* ***Risk Alert:*** Your family history shows a pattern of Type 2 Diabetes, which may suggest a higher genetic predisposition for this condition.
* ***Preventive Measures:*** Maintaining a healthy weight, engaging in regular physical activity (like 30 minutes of walking most days), and avoiding smoking are general habits that support overall metabolic health.
* ***Precautions to Discuss with a Doctor:*** You may want to talk to your doctor about regular blood sugar screenings (like an A1C test) earlier than typically recommended.
* ***General Food Diet:*** A diet that supports stable blood sugar often includes whole grains, lean proteins (chicken, fish, beans), and plenty of vegetables. Limiting sugary drinks, refined carbohydrates, and processed foods is generally advised.
* ***Disclaimer:*** This is not medical advice. Please consult a qualified healthcare professional for personalized guidance.

**SNAPSHOTS:**

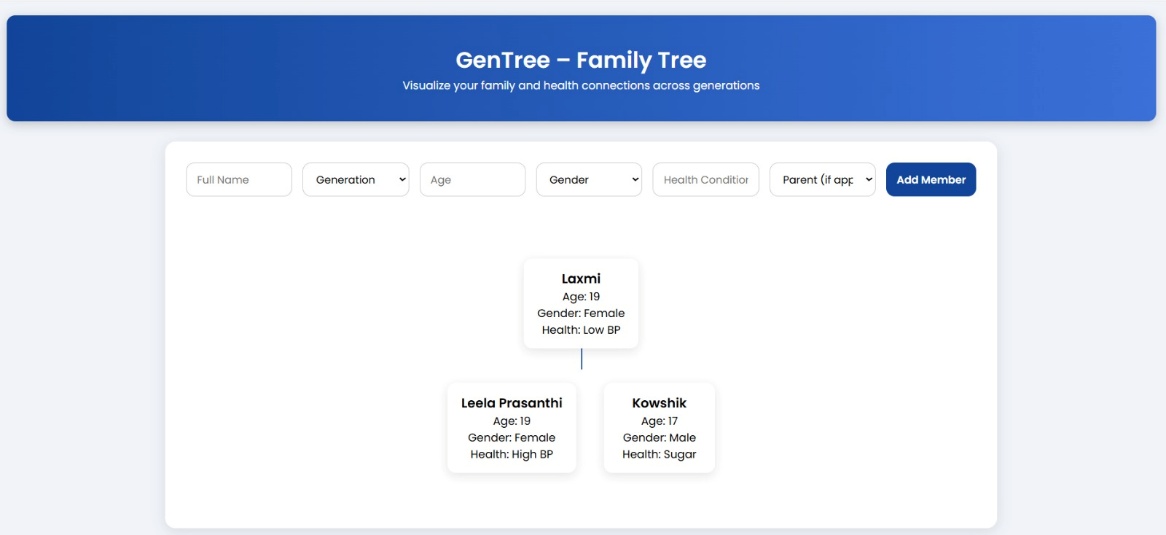
1. Login Page



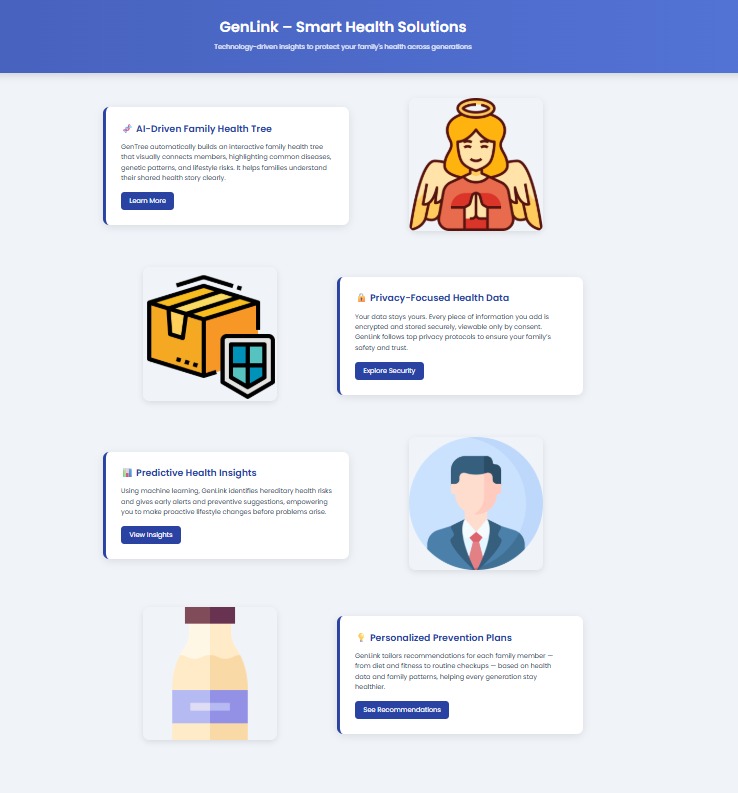
1. Personal & Family Details



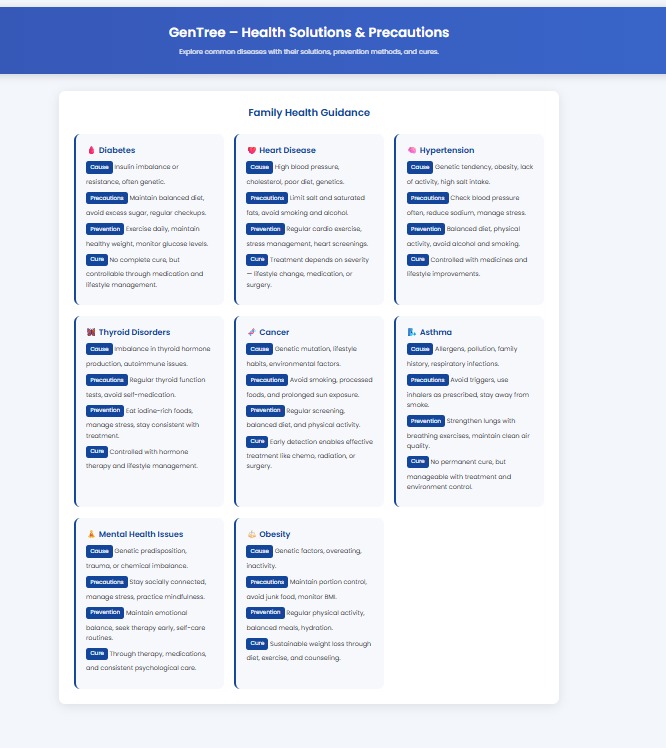
1. Family Tree



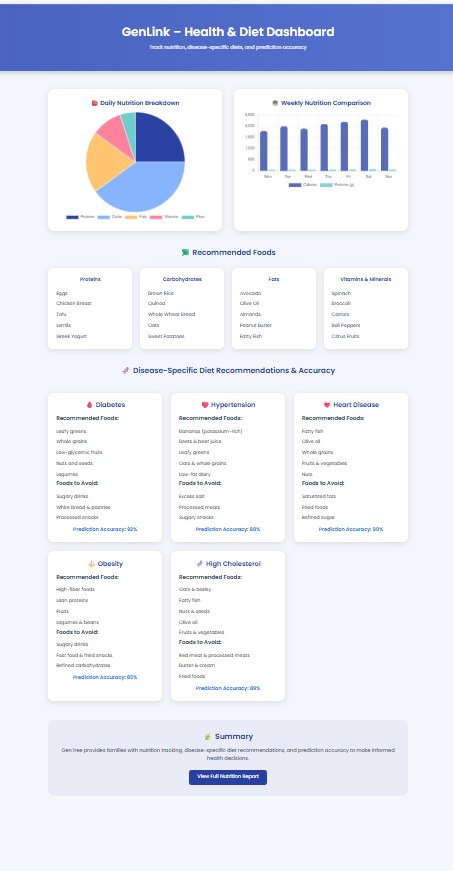
1. Smart Health Solutions



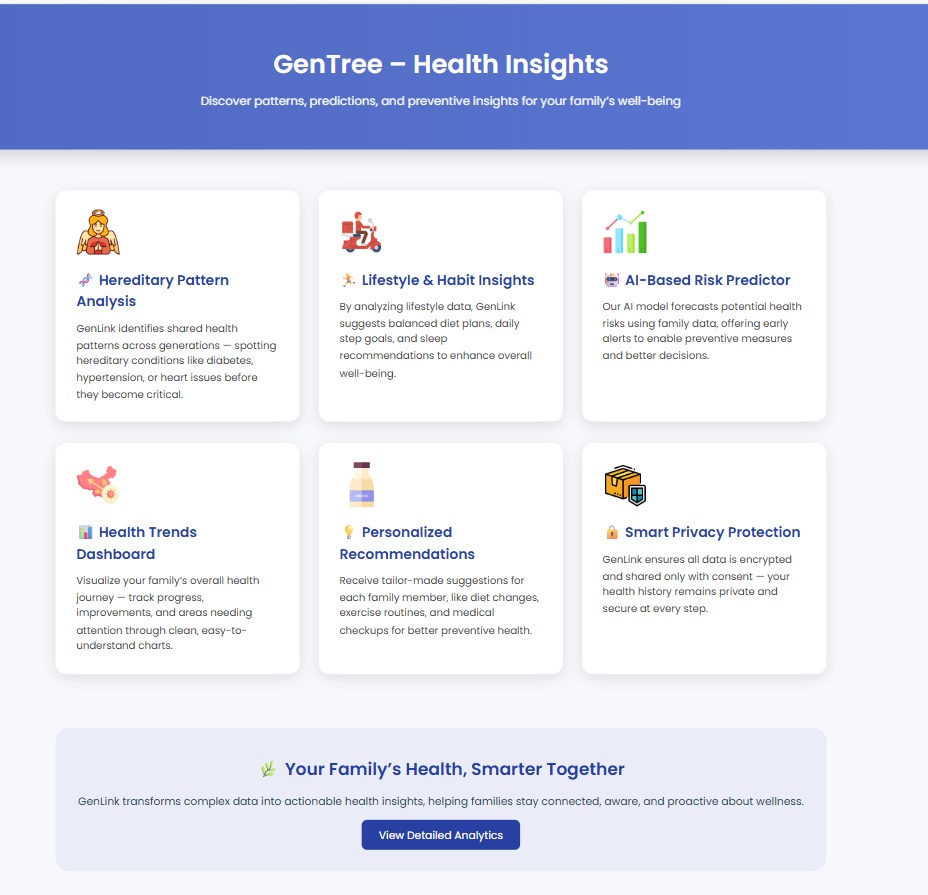
1. Health Solutions & Precautions



1. Health & Diet Dashboard



1. Health Insights



1. Health Dashboard



**DISADVANTAGES:**

* Inapplicability to Non- Traditional Families.
  + Adoption.
  + Complex Family Structure.
* Psychological and Behavioral Barriers.
* Data Accuracy.

**ADVANTAGES:**

* Sense of Control.
* Reduce Health Anxiety.
* Reduces Long-term Healthcare Costs.
* Efficiency & Better Outcomes for Healthcare Systems.
* Create a Portable and Digital Health Legacy.
* Actionable Foresight.

**REFERENCES:**

* WHO – World Health Organization- International Organization
  + <https://www.who.int/>
* CDC – Centres for Disease Control and Prevention
  + <https://www.cdc.gov/>
* WebMD – Better Information. Better Health.
  + <https://www.webmd.com/>