Veer Narmad South Gujarat University, Surat.

Department of Information and Communication Technology

M.Sc (Information Technology)

5 Year Integrated Course

Year 2024-2025

PROJECT REPORT

B.Sc. (Information Technology) 6th Semester

Niramaya – Health AI (Symptoms Checker)

Guided By:

No.	Name
1.	Mrs. Rupal Panchal

Submitted By:

Seat No.	Name	SPID
171	Harsh Sarvaiya	2022026307
193	Naitik Sutariya	2022026329
207	Kris Vaholiya	2022026346

Re-Accredited 'B++' 2.86 CGPA by NAAC
VEER NARMAD SOUTH GUJARAT UNIVERSITY
University Campus, Udhna-Magdalla Road, SURAT - 395 007, Gujarat, India.
વીર નર્મદ દક્ષિણ ગુજરાત યુનિવર્સિટી
યુનિવર્સિટી કેમ્પસ, ઉધના-મગદલ્લા રોડ, સુરત - ૩૯૫ ૦૦૭, ગુજરાત, ભારત.
Tel : +91 - 261 - 2227141 to 2227146, Toll Free : 1800 2333 011, Digital Helpline No.- 0261 2388888
E-mail : info@vnsgu.ac.in, Website : www.vnsgu.ac.in

Department of Information and Communication Technology

M.Sc.(Information Technology) Programme

Certificate

This is to certify that Mr./Ms. **Sutariya Naitik Dineshbhai** with exam seat number **193** and SPID **2022026329** has worked on his/her project work entitled as **Niramaya – Health AI (Symptoms Checker)** at **Department of ICT** as a partial fulfillment of requirement of B.Sc. (Information Technology) - 6th Semester, during the academic year 2024-2025.

Date: 22/4/2025

Place: Department of ICT, VNSGU, Surat

Internal Project Guide

BSc(IT) 6th Semester

Department of ICT

VNSGU, Surat

Head of the Department

Department of ICT

VNSGU, Surat

PROJECT OF M.Sc. I.T. PROGRAMME
VEER NARMAD SOUTH GUJARAT UNIVERSITY
Academic Year:
Approved By :
EXAMINERS

3

Index

1. Introduction

Niramaya - Health-AI is an innovative, AI-powered platform designed to empower users with reliable health information, symptom analysis, and personalized disease prediction—all through an intuitive digital interface.

1.1 Scope

Niramaya - Health-AI (Symptoms Checker) is designed as a comprehensive health advisory platform that leverages artificial intelligence to support users in understanding potential health conditions based on their reported symptoms and health-related queries. The system's scope includes:

- **Health Query Resolution:** Integrating NLP models (such as BioGPT and Gemini) to answer general health queries with contextually relevant information.
- **Symptom-Based Disease Prediction:** Utilizing machine learning—specifically logistic regression models (LRv1 and LRv2)—to predict potential diseases and provide preventive measures.
- **Personalized User Management:** A family-centric user system similar to a Netflix account structure, allowing a single primary user to create multiple member profiles.
- **Research Integration:** Displaying curated health research papers on the home page to educate users further.
- **Interactive Communication:** Offering a chat-like interface where users can view past interactions, receive new responses from chosen AI models, and update member profiles as needed.

This scope ensures that users not only get accurate and timely health information but also have an engaging platform for managing their health history and family profiles.

1.2 Objective

The primary objectives of *Niramaya* are:

- **Accurate Health Guidance:** To provide users with reliable answers to their health-related questions by using sophisticated AI models that interpret natural language queries.
- **Symptom Analysis:** To enable users to enter their symptoms and receive quick, evidence-based predictions regarding potential diseases. This helps in early awareness and timely action.
- **Preventive Recommendations:** To assist users with precautionary measures and explanations for the predicted diseases, thereby promoting proactive health management.
- **User Personalization:** To allow seamless management of individual family member profiles under a single account, ensuring personalized interaction and tracking of health queries.
- **Enhanced Accessibility:** To deliver a user-friendly and accessible web interface that can be utilized across various operating systems and devices, ensuring healthcare information is always within reach.

These objectives guide the system's design, implementation, and continuous improvement, ensuring its value as an accessible health resource.

1.3 Existing System

Traditionally, individuals seeking health guidance have relied on methods such as:

- **Internet Searches:** Utilizing search engines like Google to look up symptoms and find information on possible conditions, often resulting in information overload and potential misdiagnoses.

- **Social Interactions:** Relying on advice from friends or family, which may not always be accurate or suitable to an individual's specific circumstances.
- **Healthcare Consultations:** Visiting clinics or doctors for professional opinions, which could be time-consuming, expensive, or inaccessible, particularly in remote regions.

These conventional approaches often lead to fragmented information, increased anxiety due to conflicting opinions, and inefficient pathways to proper care. There is a significant gap between available health information and the need for a quick, reliable, and personalized advisory service.

1.4 Purpose of System

The purpose of *Niramaya* is to bridge the gap between the diverse sources of health information and the immediate needs of users by offering an integrated, AI-powered platform that:

- **Delivers Accurate Health Answers:** By combining state-of-the-art natural language processing (NLP) and machine learning techniques, the system provides users with tailored answers and predictions.
- **Empowers Users with Self-Diagnosis Tools:** Enables individuals to perform an initial self-check via symptom analysis, reducing anxiety and offering guidance on whether to seek further medical consultation.
- **Facilitates Informed Health Decisions:** Provides not only the likely medical condition but also important precautions and background information, which helps users make informed health decisions.
- **Personalizes the Health Experience:** Offers a family-based account system where multiple profiles can be managed, supporting personalized data tracking and query history for each individual.

- **Encourages Continuous Learning:** Includes access to research papers and additional resources that support user education and proactive health management.

In essence, the system's purpose is to democratize health information, ensuring it is both reliable and accessible, thus promoting overall well-being within the community.

1.5 Project Profile

Project Title	Niramaya – Health AI
Project Definition	Niramaya is an AI-powered health assistant that helps users check symptoms, get disease predictions, and receive reliable medical responses in real-time.
Server Side Technologies	HTML, CSS, JavaScript
Client Side Technologies	Python with Flask
Database	PostgreSQL
Project Category	Web Application
Guided By	Dr. Rupal Panchal
Developed By	Naitik Sutariya Kris Vaholiya Harsh Sarvaiya

2. System Environment

2.1 Hardware Requirements

Processor	Intel Core i3 (5th generation or later) or equivalent AMD processor
RAM	At least 4 GB of RAM is required
Input Devices	Keyboard, Mouse
Storage	256 GB
Internet Connection	A stable broadband connection (minimum 512 Kbps)

2.2 Software Requirements

Operating System	Windows, IoS, Linux, Android
Web Browser	Google Chrome, Safari, Firefox, Edge
Server Side Technologies	Python with Flask Machine Learning Libraries : • Scikit-learn • Transformers (Hugging Face)
Client Side Technologies	HTML, CSS, JavaScript
Database	PostgreSQL
Development and Testing Tools	• VS Code • Postman • pgAdmin
Server	Localhost

3. System Requirement Specification

3.1 Functional Requirements

These requirements define what the system should do. The main functionalities for *Niramaya - Health-AI (Symptoms Checker)* include:

- **User Account Management:**
    - **Registration & Login:** Users must be able to create an account, register using email/password, and securely log in.
    - **Family Account Structure:** A single account can support multiple family member profiles, allowing users to add, update, and remove member details.
- **Health Query and Symptom Input:**
    - **Question Input:** Users can submit health-related questions that the system processes using advanced NLP models (e.g., BioGPT and Gemini).
    - **Symptom Entry:** Users can input their symptoms to receive predictions on possible diseases.
- **Response Generation:**
    - **Disease Prediction:** Based on the entered symptoms, the system utilizes a logistic regression model (LRv1) to provide a probable disease prediction along with a brief description.
    - **Precautionary Guidance:** A secondary model (LRv2) processes the input to generate recommended precautions and further details regarding the predicted disease.
    - **Conversational Interaction:** The system provides responses in a chat-like format, allowing users to have a conversational experience with the AI.

- **Chat History Management:**
    - **History Tracking:** Users can view a log of previous questions, responses, and other interactions.
    - **History Deletion:** Users can delete specific entries from their chat history for privacy or clarity.
- **Content and Research Integration:**
    - **Resource Access:** The homepage offers links to curated health research articles and papers, enabling users to gain further insight and background information.
- **Profile and Member Management:**
    - **View and Update Profiles:** Users can view and update both primary account and member profiles.
    - **Switch Member Accounts:** Quick switching between family member profiles to view personalized health information.
- **Error Handling and Notifications:**
    - **Input Validation:** The system validates user inputs and ensures proper data formats (e.g., email, required fields).
    - **User Notifications:** Notifications and alert messages are provided upon successful actions (e.g., account creation, successful submission) or in case of errors (e.g., incorrect password, data submission failures).

3.2 Non-Functional Requirements

These requirements describe how the system performs its functions rather than what it does. Non-functional requirements for *Niramaya* ensure the system's quality, user satisfaction, and scalability:

- **Performance:**
    - **Response Time:** The system should respond to user queries within a few seconds, ensuring minimal latency even when processing machine learning models.
    - **Scalability:** The application architecture should handle a growing number of concurrent users and family accounts without degradation in performance.
- **Usability:**
    - **User-Friendly Interface:** The web application must have an intuitive design with clear navigation, ensuring ease of use for both tech-savvy and non-technical users.
    - **Accessibility:** The application should be compatible with modern web browsers and accessible on various devices (desktops, tablets, mobile phones).
- **Reliability and Availability:**
    - **Uptime:** The system should have high availability, ensuring minimal downtime and reliable access to services.
    - **Error Recovery:** Robust error-handling mechanisms must be in place to quickly recover from unexpected failures or errors.
- **Security:**
    - **Data Protection:** Ensure sensitive user data (e.g., personal details, health queries) is securely stored using encryption and best practices in database management.
    - **User Authentication:** Strong authentication and authorization protocols must be enforced to prevent unauthorized access.

- **Maintainability:**
    - **Modular Architecture:** The system should be designed using modular components to facilitate easier updates, bug fixes, and feature enhancements.
    - **Documentation:** Comprehensive documentation must be maintained for both developers and end-users to support future maintenance and upgrades.
- **Compliance:**
    - **Data Privacy Standards:** The system must adhere to relevant data protection and privacy regulations (such as GDPR for European users or similar local regulations) to ensure user trust and legal compliance.
- **Internationalization and Localization:**
    - Although primarily designed for a specific user base, the system should be easily adaptable for multiple languages and regional settings if needed.

4. System Planning

4.1 Feasibility Study

The feasibility study examines whether *Niramaya - Health-AI (Symptoms Checker)* can be successfully implemented from multiple perspectives:

- **Technical Feasibility:**
  The project utilizes established, open-source technologies such as Python with Flask for the backend, PostgreSQL for database management, and robust AI/ML libraries (Scikit-learn and Transformers). These tools have strong community support and extensive documentation, ensuring that the technical challenges (such as AI model integration and scalable web application development) are manageable.

- **Economic Feasibility:**
  Leveraging open-source libraries minimizes development costs. Standard hardware and cloud infrastructure can efficiently support the application, making it a cost-effective solution. Moreover, the potential reduction in unnecessary healthcare visits through early symptom screening provides an economic benefit that further justifies the investment.

- **Operational Feasibility:**
  With an intuitive, user-friendly interface and clear account management (including a family account model), the system is designed to be accessible to a broad user base—even to those who are not technically inclined. The operational procedures, including user authentication, symptom input, and real-time response generation, ensure that day-to-day usage remains efficient and reliable.

- **Legal and Social Feasibility:**
  Niramaya emphasizes secure data management and privacy to comply with data protection regulations. The system's design reinforces user trust by safeguarding sensitive health information, thereby meeting both legal obligations and societal expectations.

4.2 Software Engineering Model

Niramaya is developed using an **Agile Software Engineering Model** to ensure flexibility and adaptability throughout the project lifecycle:

- **Iterative Development:**
  The project is broken down into multiple short development cycles (sprints), allowing for continuous improvement and regular feedback. This structure helps in quickly addressing any issues or changing requirements.
- **Rapid Prototyping:**
  Early prototypes are developed to validate core concepts such as health query processing and symptom analysis. These prototypes facilitate early user testing, enabling designers and developers to fine-tune functionality before full-scale development begins.
- **Collaborative Approach:**
  Regular team meetings, code reviews, and sprint planning sessions ensure transparent communication among team members. This collaborative structure helps align development activities with project objectives, ensuring that all components of the system integrate seamlessly.
- **Quality Assurance:**
  Continuous integration and testing (unit testing, integration testing, and system testing) are central to the development process. Automated tests and manual QA cycles ensure that each module meets quality standards before it is integrated into the overall system.

4.3 Risk Analysis

A comprehensive risk analysis was conducted to identify potential challenges and establish mitigation strategies:

- **Technical Risks:**
    - *Integration Challenges:* Combining different AI models (like BioGPT, Gemini, and the LR models) might introduce compatibility issues.
        - **Mitigation:** Early integration tests and iterative interface refinement will help reduce these risks.
    - *Performance Bottlenecks:* Machine learning models may demand higher computational resources, impacting system responsiveness.
        - **Mitigation:** Code optimization, caching strategies, and scalable deployment (potentially on cloud environments) will address performance issues.
- **Operational Risks:**
    - *User Adoption:* New users may experience a learning curve when interacting with the AI-driven interface.
        - **Mitigation:** Develop comprehensive user guides, intuitive UI design, and provide onboarding tutorials.
    - *Data Privacy:* Handling sensitive personal and health information poses the risk of data breaches.
        - **Mitigation:** Implement robust security measures, including encryption of sensitive data, secure authentication, and regular security audits.
- **Project Management Risks:**
    - *Timeline Delays:* Unanticipated technical or integration challenges might extend the development schedule.

        - **Mitigation:** Incorporate buffer time in the schedule and adopt agile practices to adapt to delays.
    - *Resource Availability:* Limited access to specialized expertise or development resources could slow progress.
        - **Mitigation:** Prioritize tasks based on impact, leverage open-source solutions, and seek external expertise as needed.

4.4 Project Schedule

The project is structured over a three-month period, divided into several key phases to ensure steady progress and timely delivery:

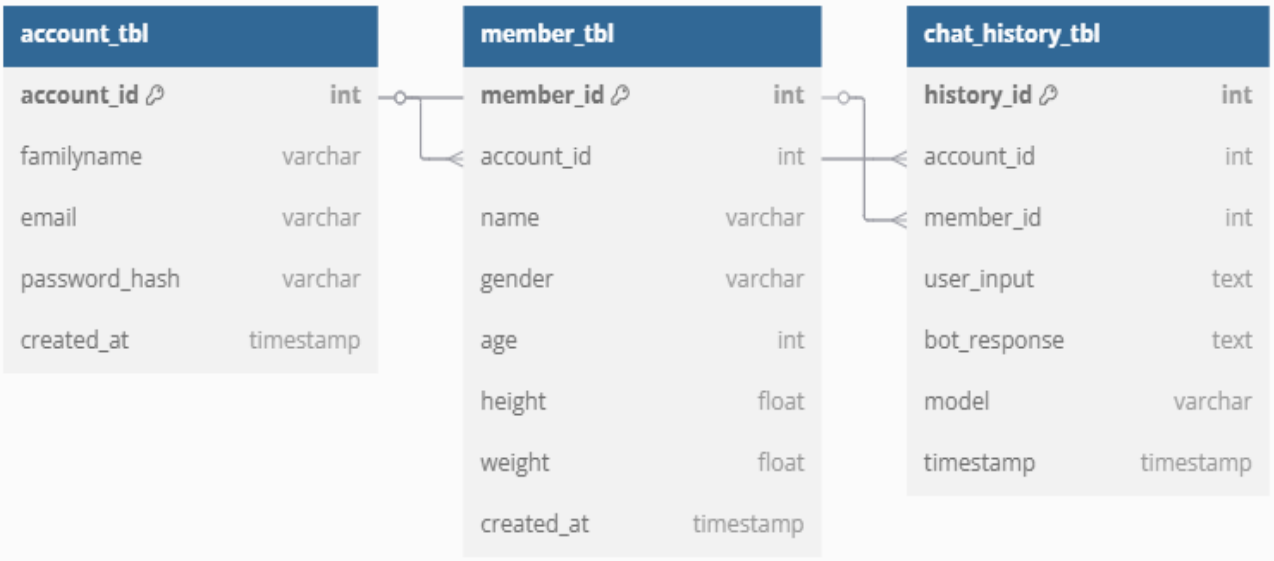
- **Phase 1: Requirement Analysis & Planning (Weeks 1-2)**
    - Finalize project scope, objectives, and system requirements.
    - Conduct initial feasibility studies and risk assessments.
    - Develop foundational documentation and project planning materials.
- **Phase 2: Design & Prototyping (Weeks 3-5)**
    - Create initial wireframes and UI mockups.
    - Develop system design diagrams (ER diagrams, UML diagrams).
    - Build and review early prototypes for core functionalities such as user authentication and basic symptom input.
- **Phase 3: Development & Implementation (Weeks 6-10)**
    - Build frontend and backend components.
    - Integrate AI models (BioGPT, Gemini, and logistic regression models) into the application.

- Implement user account management, family profiles, and chat history functionalities.
- Conduct continuous integration and unit testing.

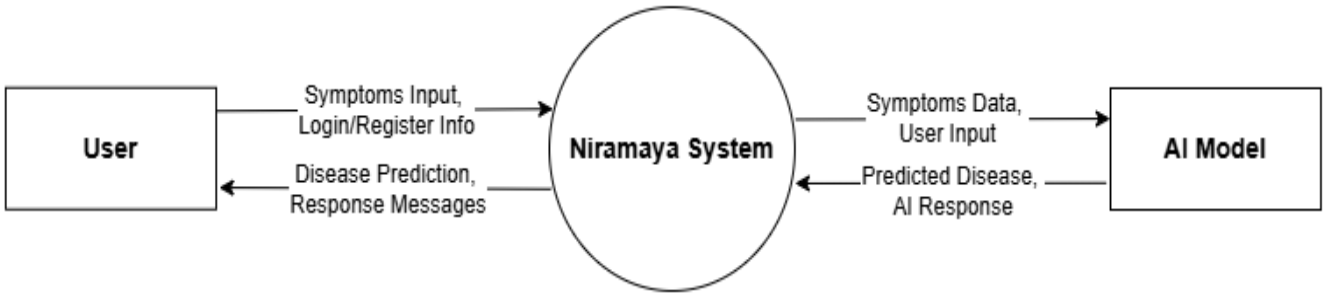
- **Phase 4: System Testing & Refinement (Weeks 11-12)**
  - Perform integration and system testing for all modules.
  - Gather user feedback and conduct usability testing.
  - Refine system components based on test results and feedback.
- **Phase 5: Deployment & Final Documentation (Week 13)**
  - Deploy the application to a production environment.
  - Finalize comprehensive project documentation.
  - Conduct a final project review meeting and prepare the project presentation.

5. System Analysis & Modeling

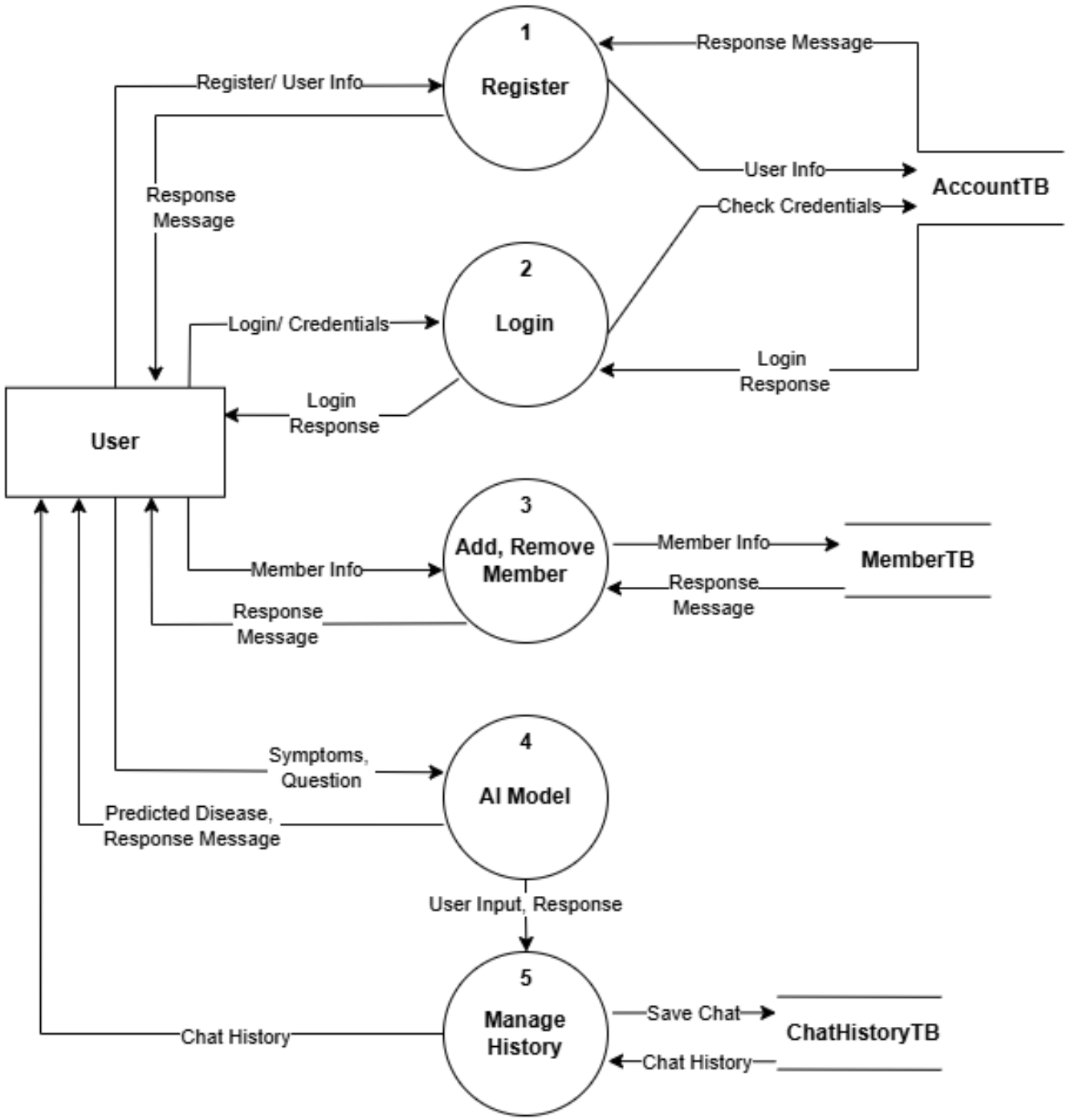
5.1. Entity-Relationship Diagram



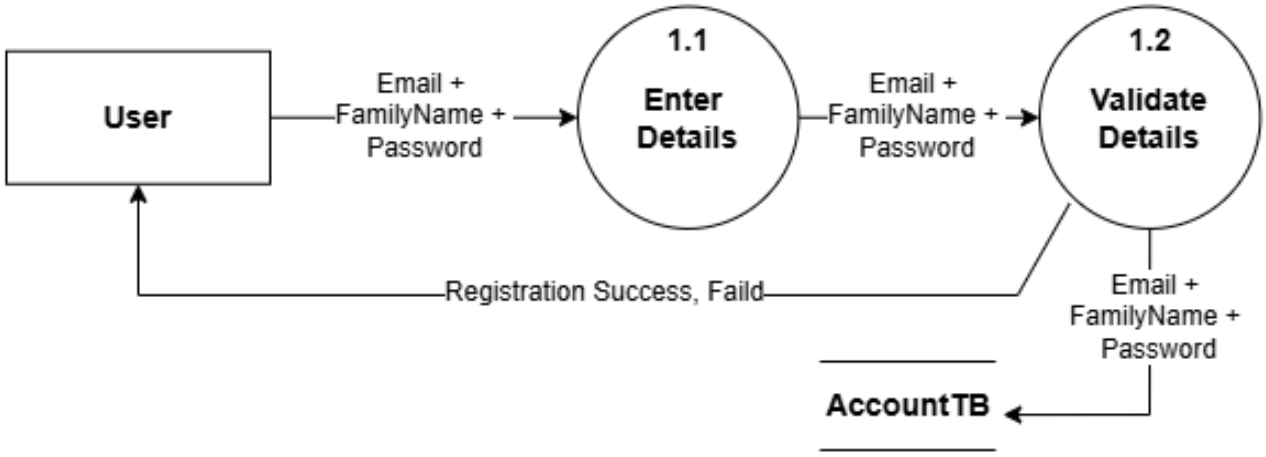
5.2. Data Flow Diagram



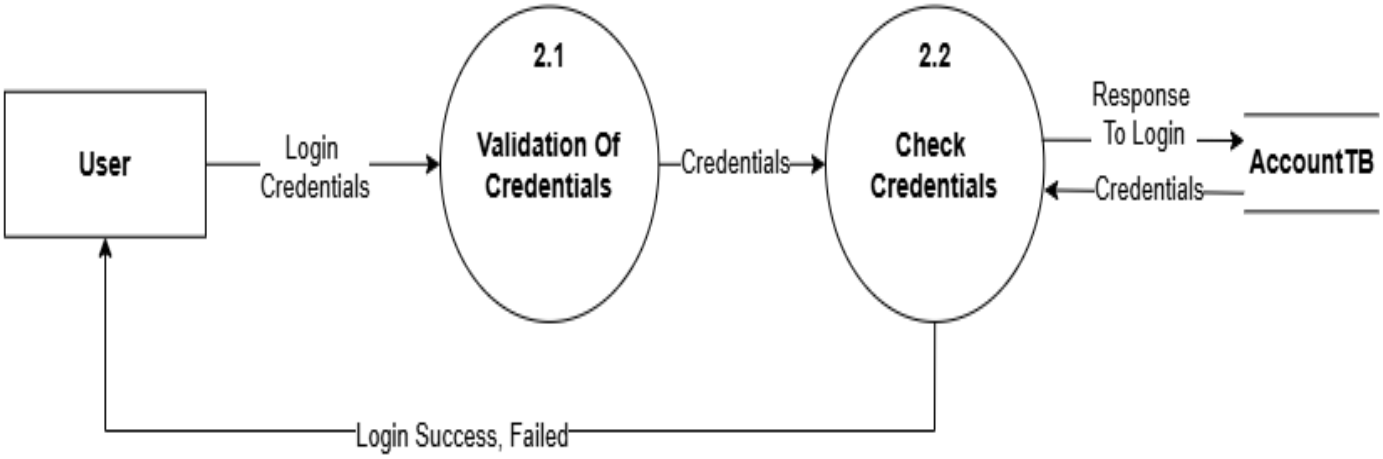
Level-0 DFD


1
Register
Response Message
Register/ User Info
User Info
AccountTB
Response
Message
Check Credentials
2
Login
Login/ Credentials
Login
Response
Login
Response
User
3
Add, Remove
Member
Member Info
MemberTB
Member Info
Response
Message
Response
Message
4
AI Model
Symptoms,
Question
Predicted Disease,
Response Message
User Input, Response
5
Manage
History
Save Chat
ChatHistoryTB
Chat History
Chat History

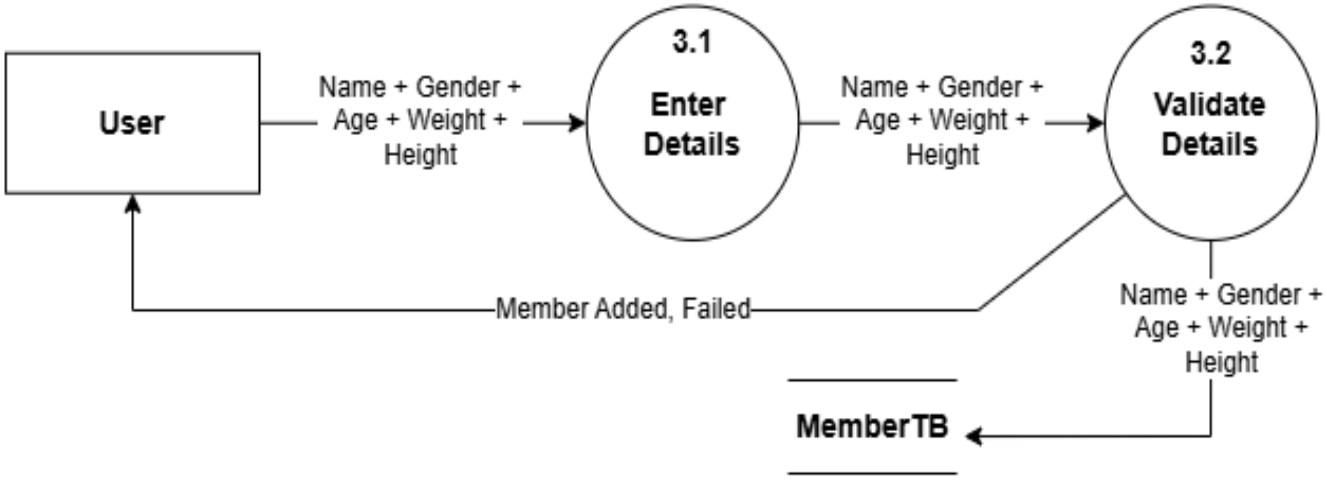
Level-1 DFD



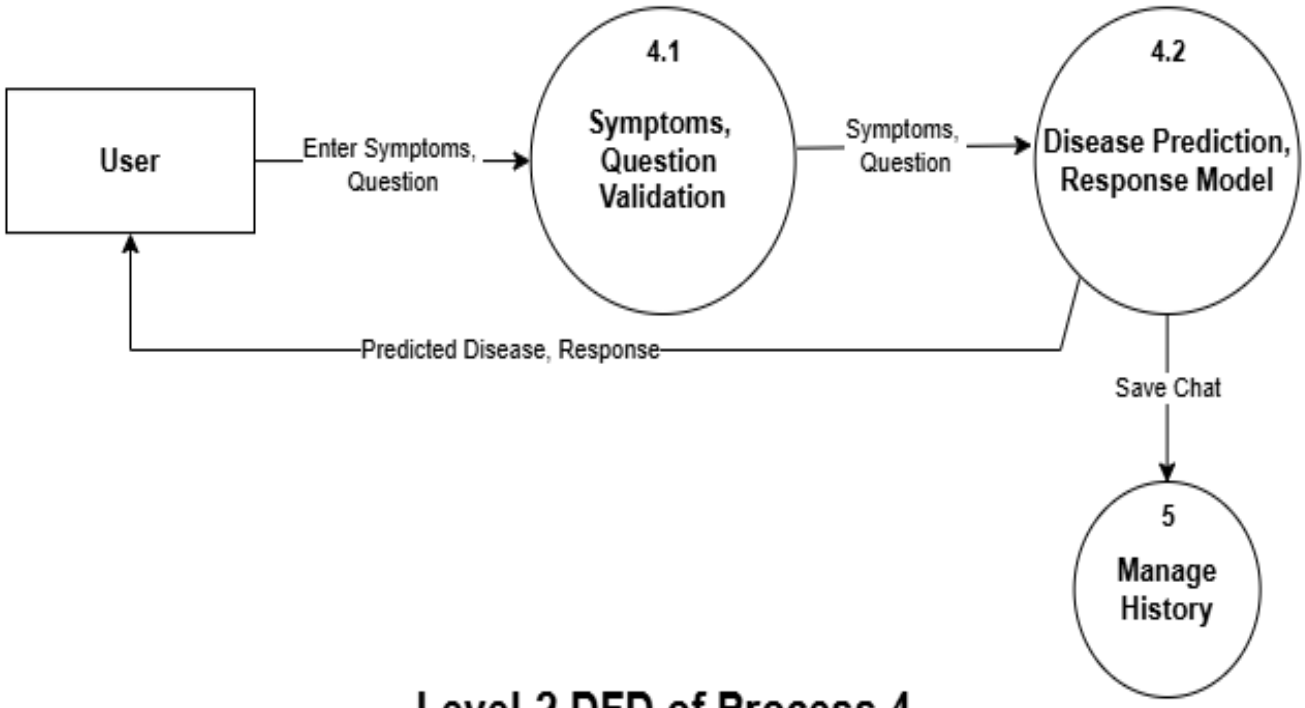
Level-2 DFD of Process 1



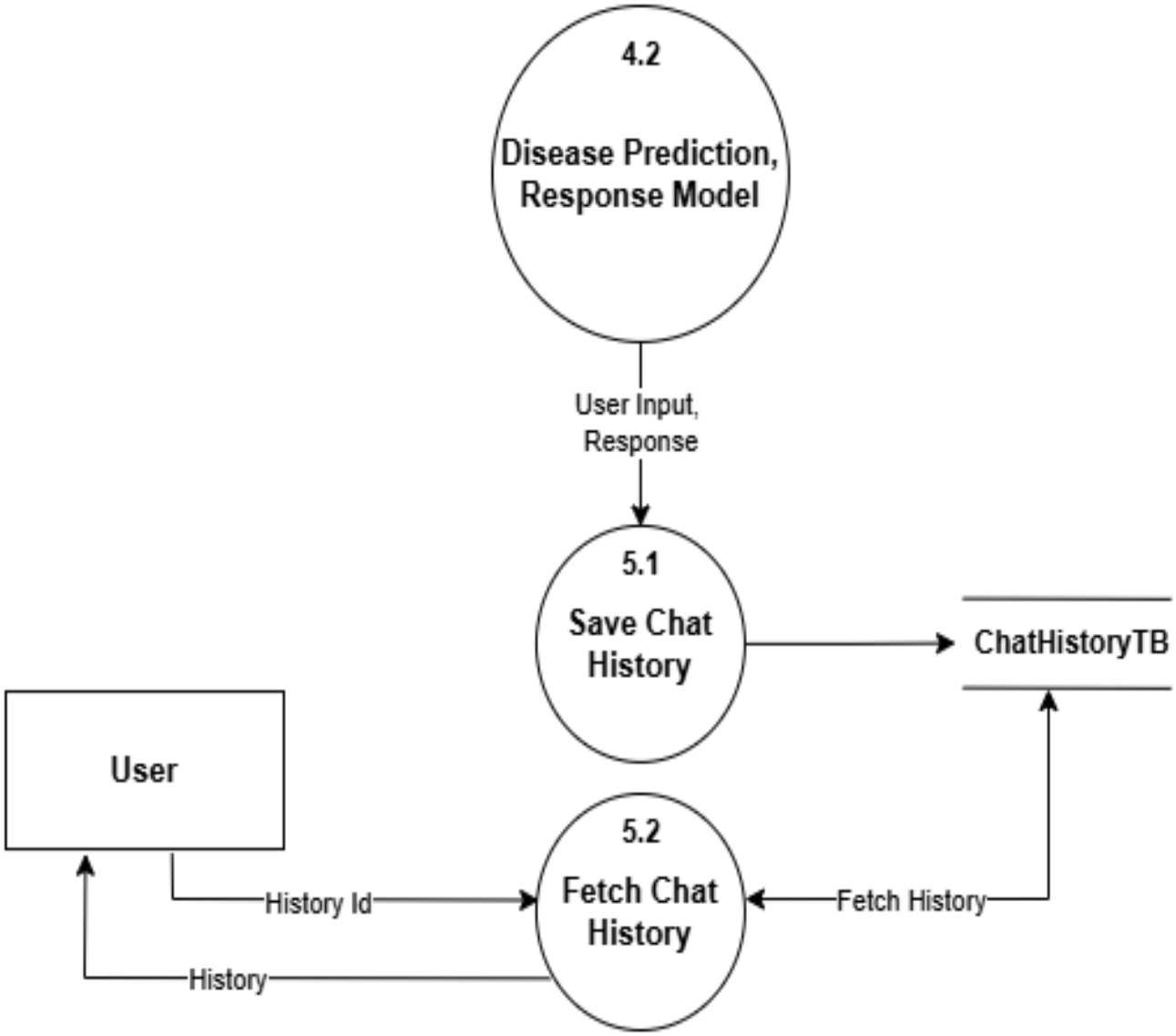
Level-2 DFD of Process 2



Level-2 DFD of Process 3



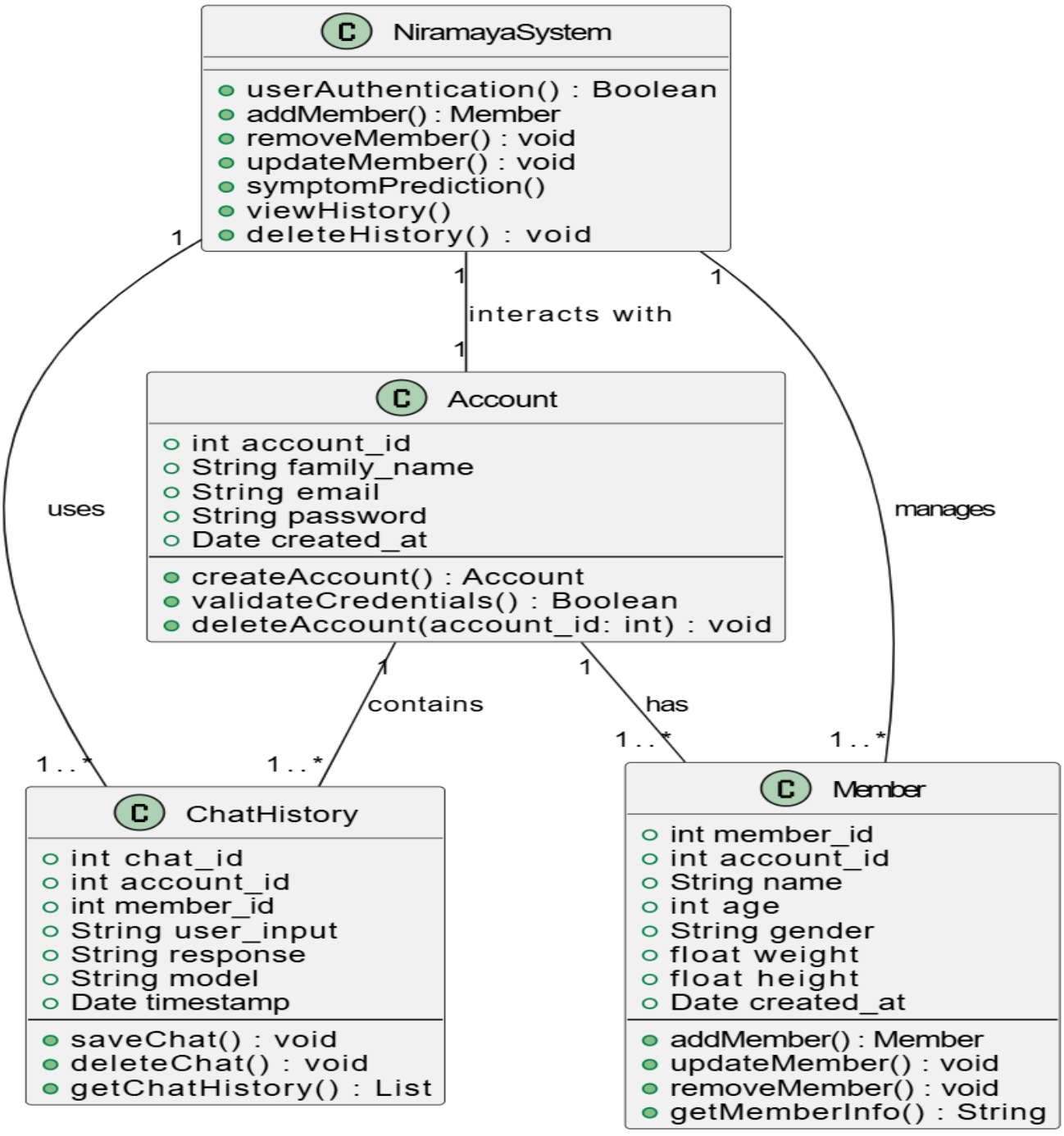
Level-2 DFD of Process 4



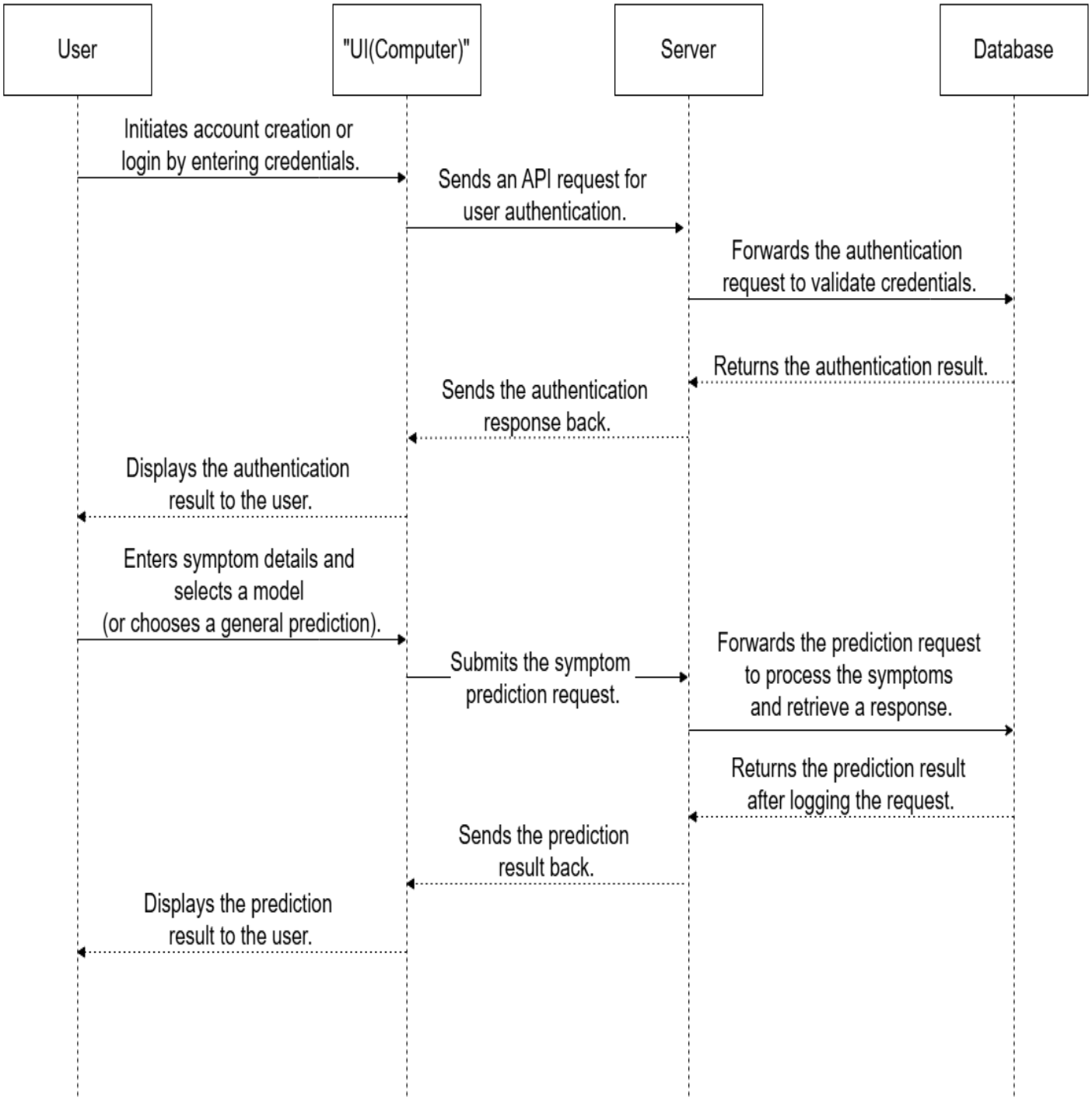
Level-2 DFD of Process 5

5.3. UML Diagram

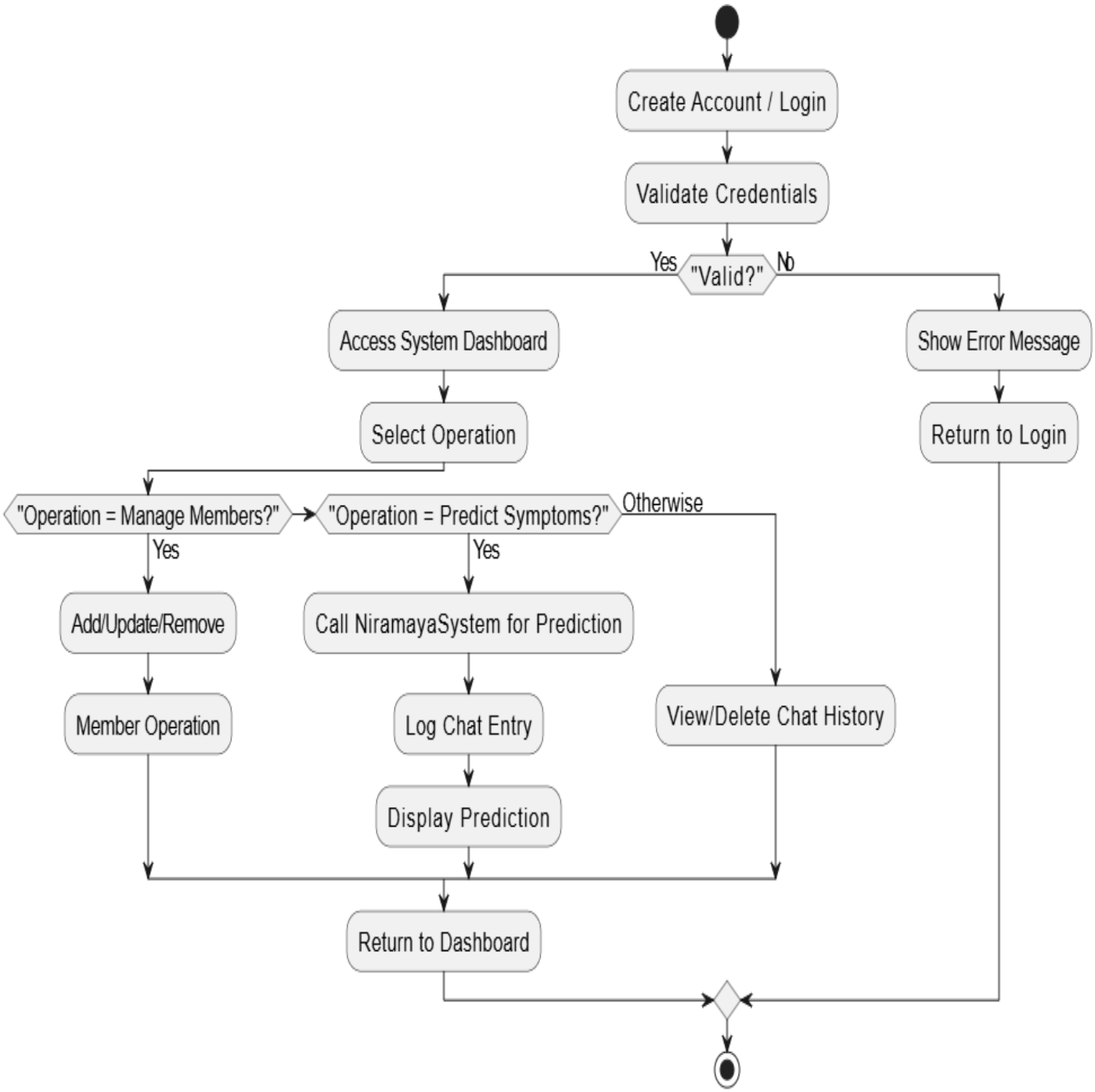
● Class Diagram



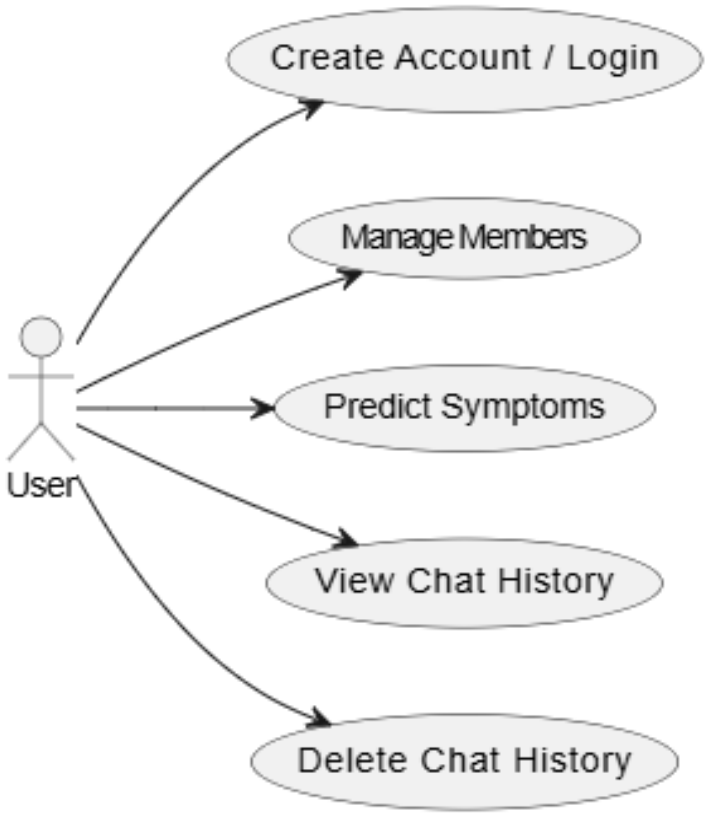
● Sequence Diagram



● Activity Diagram



● Use Case Diagram



6. System Design

6.1. Database Design

➢ Account Table :

Column Name	Data Type	Constraints	Description
account_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique account ID.
email	VARCHAR(150)	UNIQUE, NOT NULL	Account email address.
password	VARCHAR(255)	NOT NULL	Encrypted password.
Family Name	VARCHAR(100)	NOT NULL	Family account name.
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Account creation timestamp.

➢ Member Table

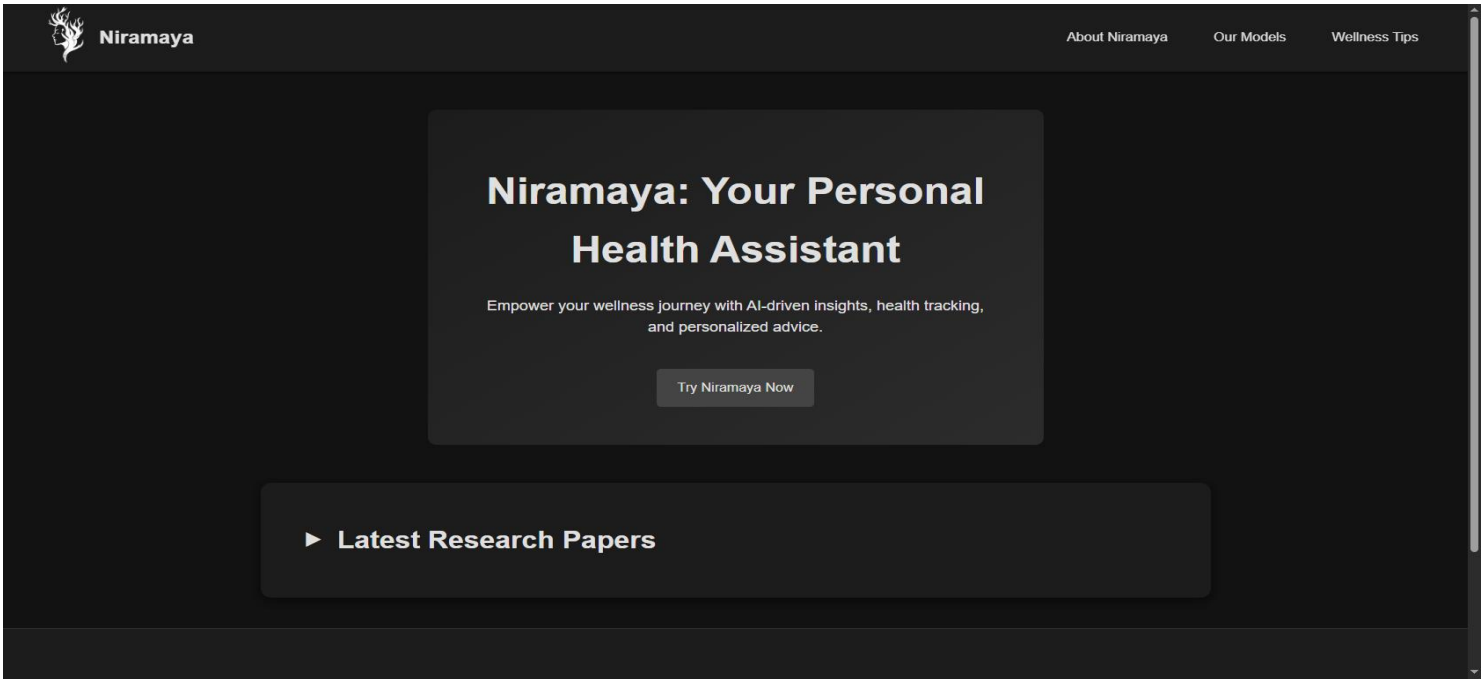
Column Name	Data Type	Constraints	Description
member_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique member ID.
account_id	INT	FOREIGN KEY REFERENCES Account(account_id)	Linked to family account.
name	VARCHAR(100)	NOT NULL	Name of the family member.
gender	ENUM('Male', 'Female', 'Other')	NOT NULL	Gender of the member.
age	INT	NOT NULL	Age of the member.
height	FLOAT	NULL	Height of the member (optional).
weight	FLOAT	NULL	Weight of the member (optional).
Created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Account creation timestamp.

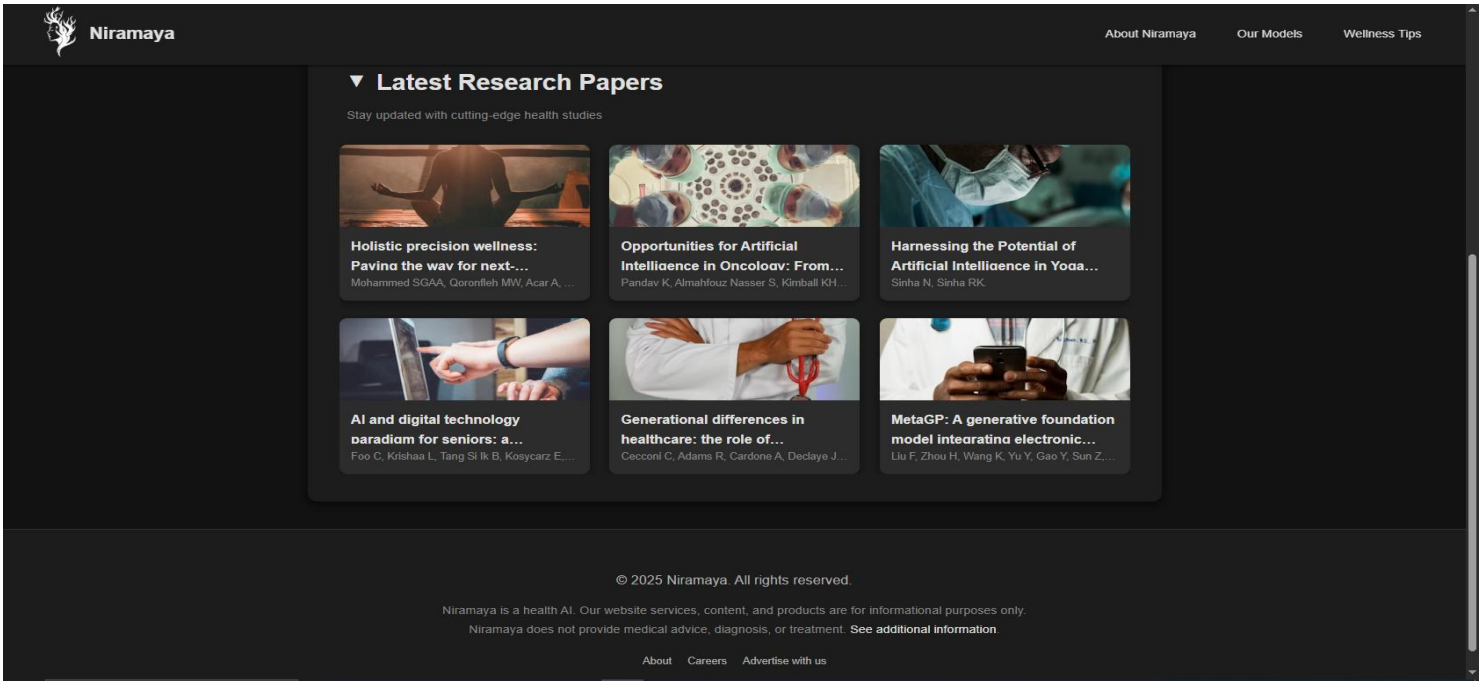
➢ Chat History Table

Column Name	Data Type	Constraints	Description
chat_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique chat ID.
member_id	INT	FOREIGN KEY REFERENCES Members(member_id)	Member asking questions.
Account_id	INT	FOREIGN KEY REFERENCES Account(account_id)	Member asking questions.
User Input	TEXT	NOT NULL	Symptom or query asked by the member.
Response	TEXT	NULL	AI-generated answer or recommendation.
Model	VARCHAR(50)	NOT NULL	Detail of Model which is Used
asked_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Timestamp when the query was made.

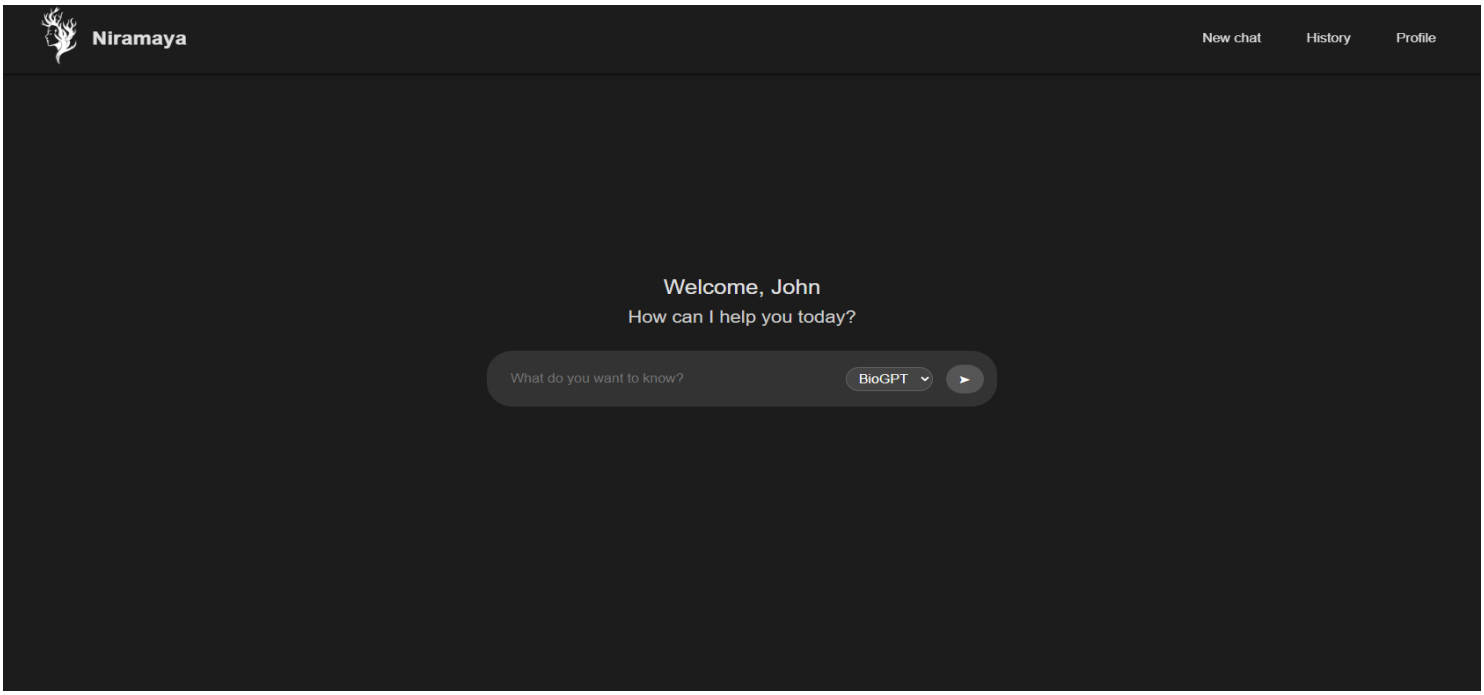
6.2. Interface Design

1. Home Page

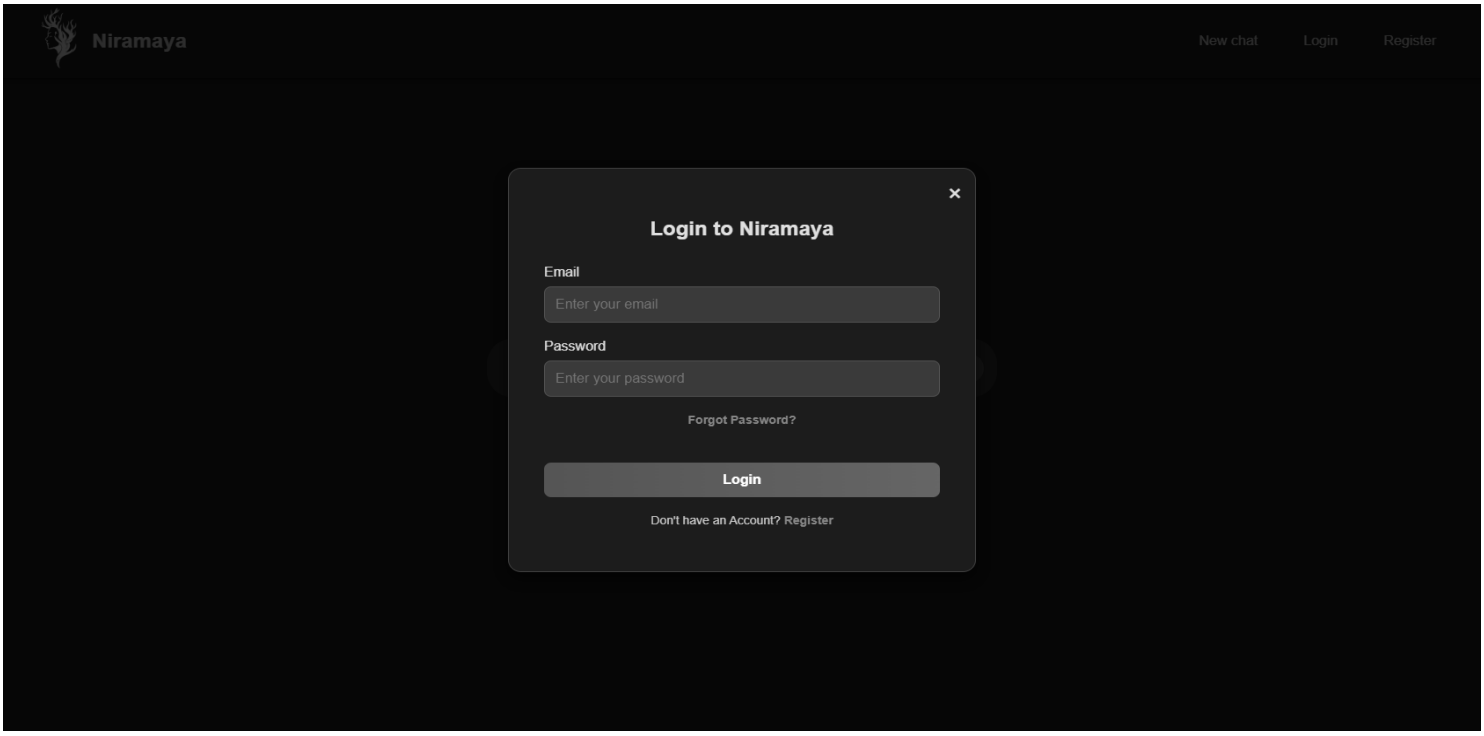




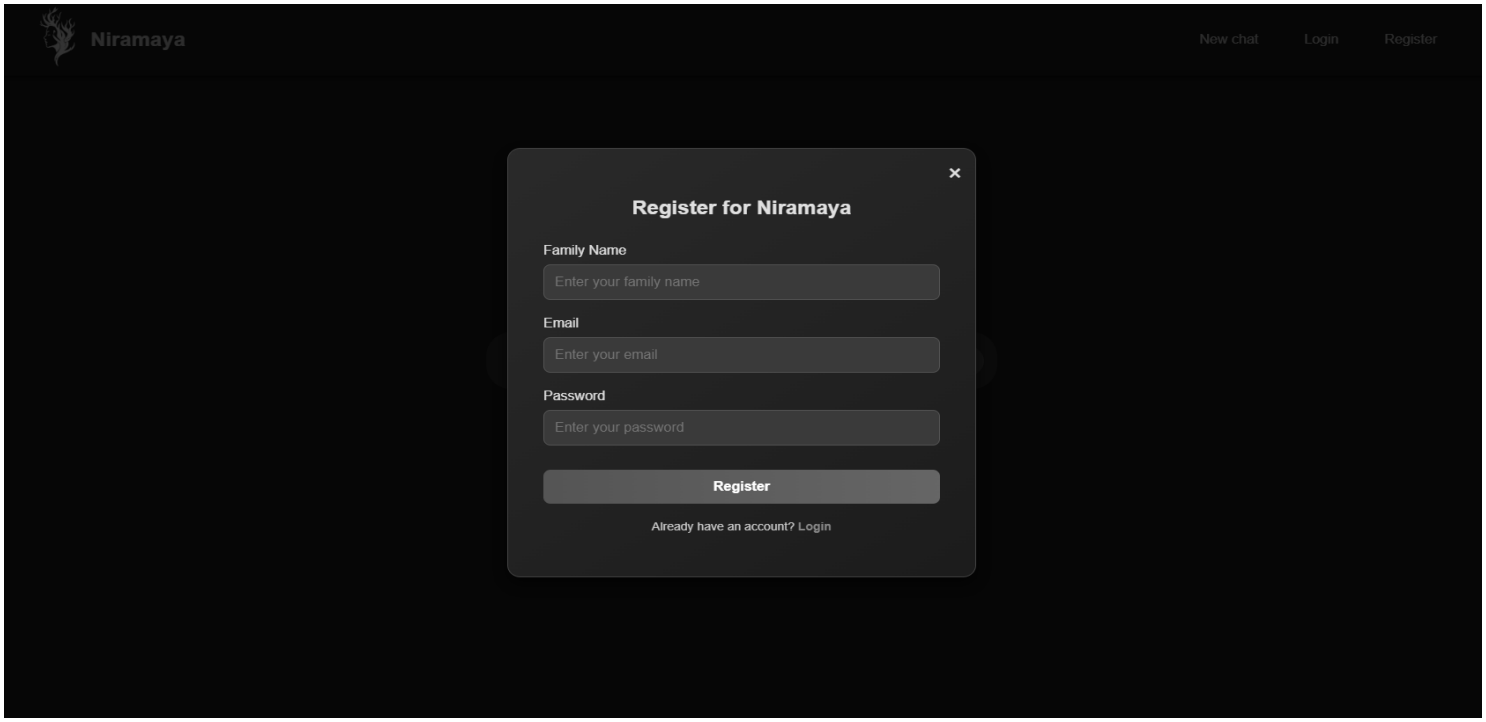
2. Niramaya Question Page



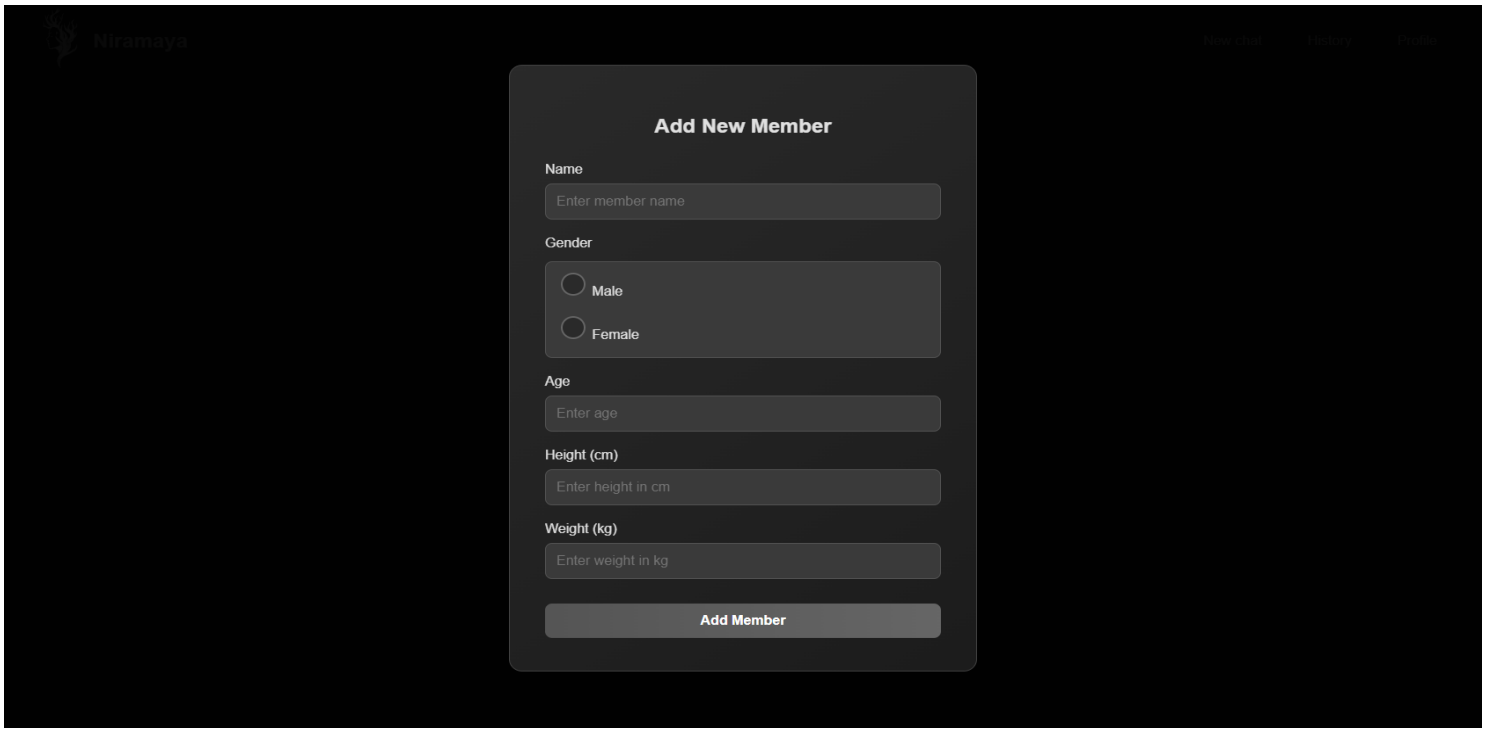
Login



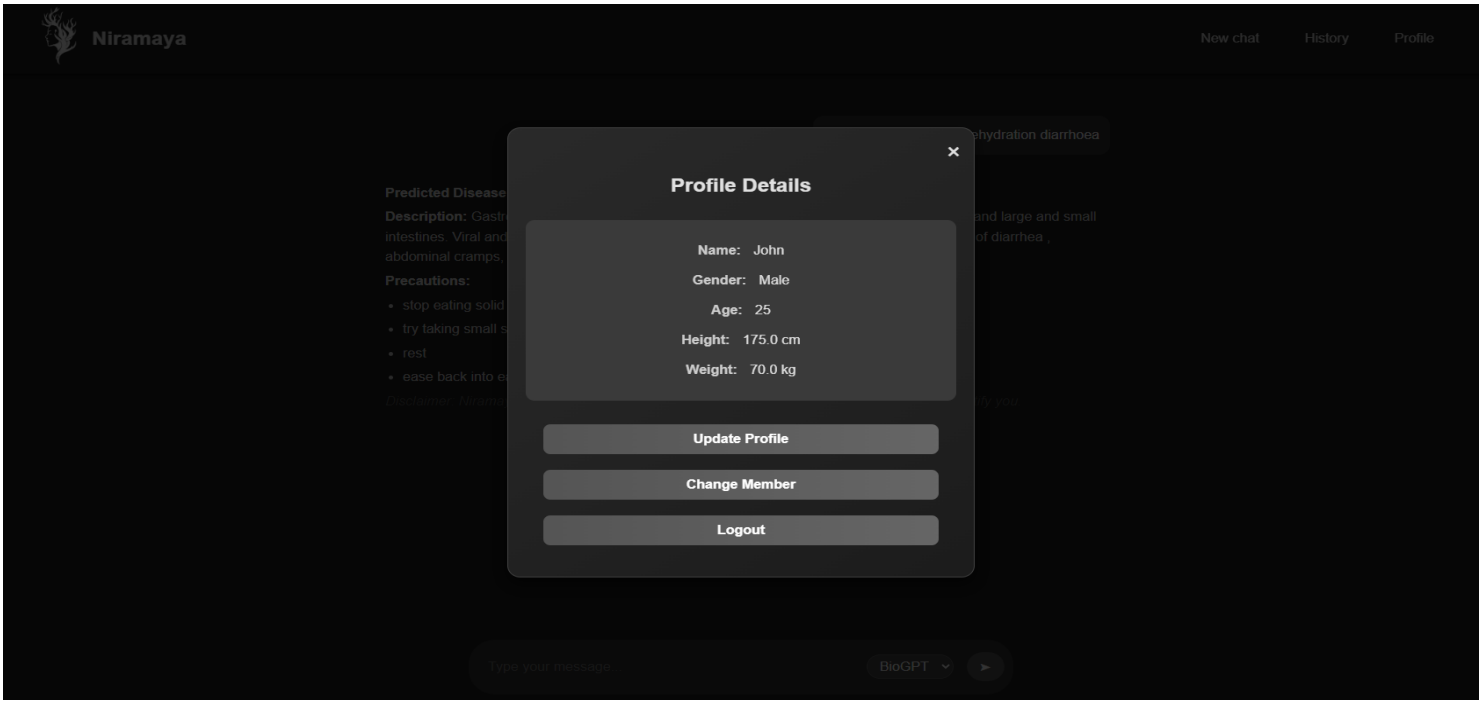
Register



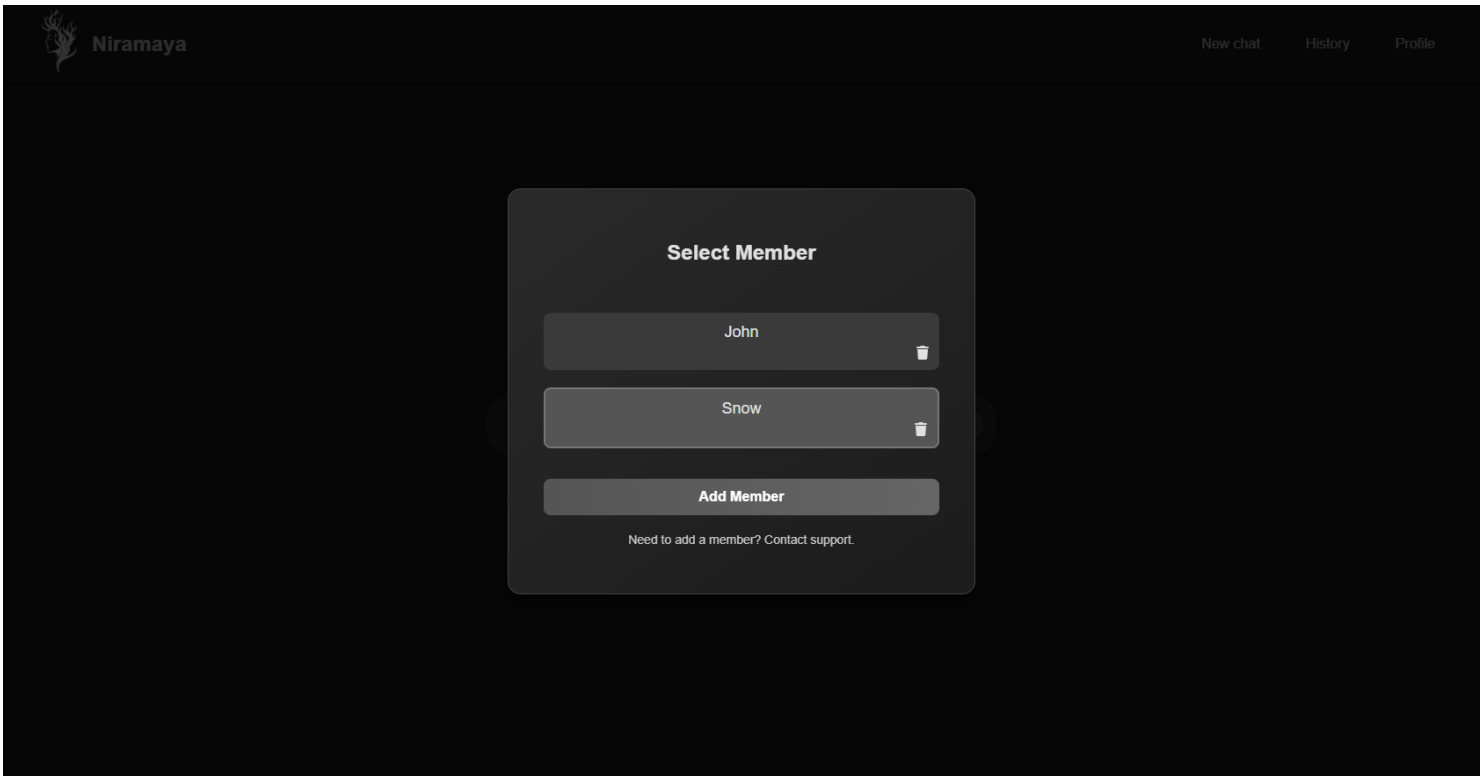
Add Member



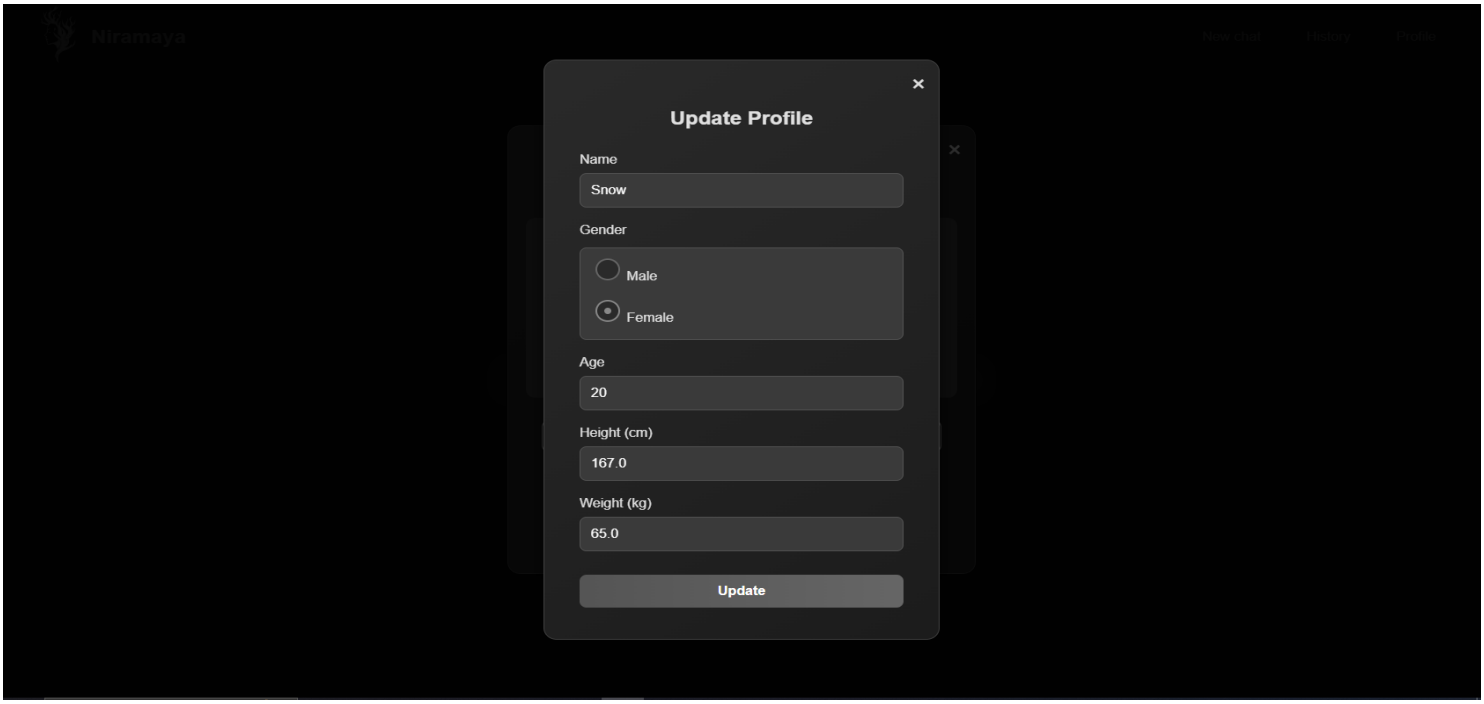
Profile



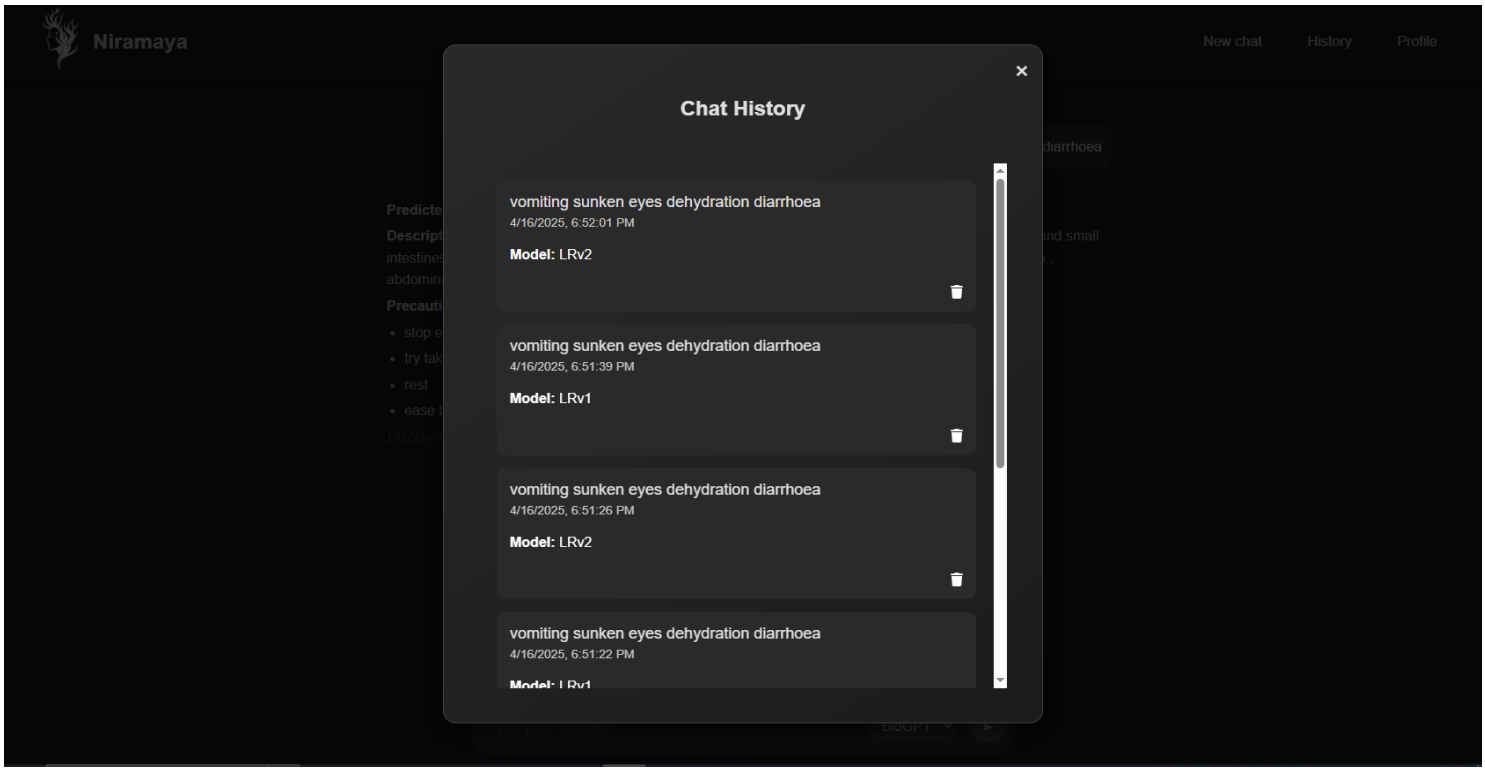
Select Member



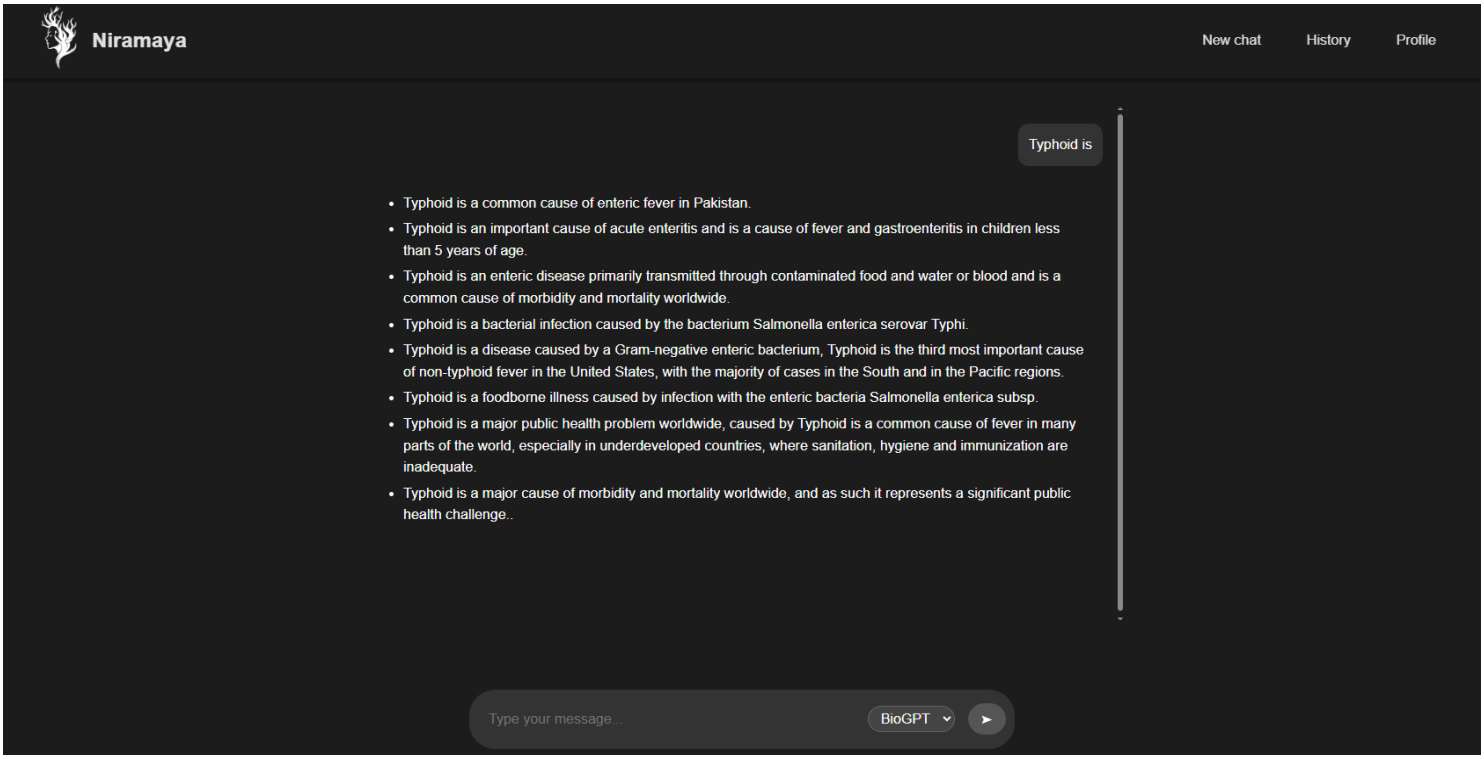
Update Profile of Memb



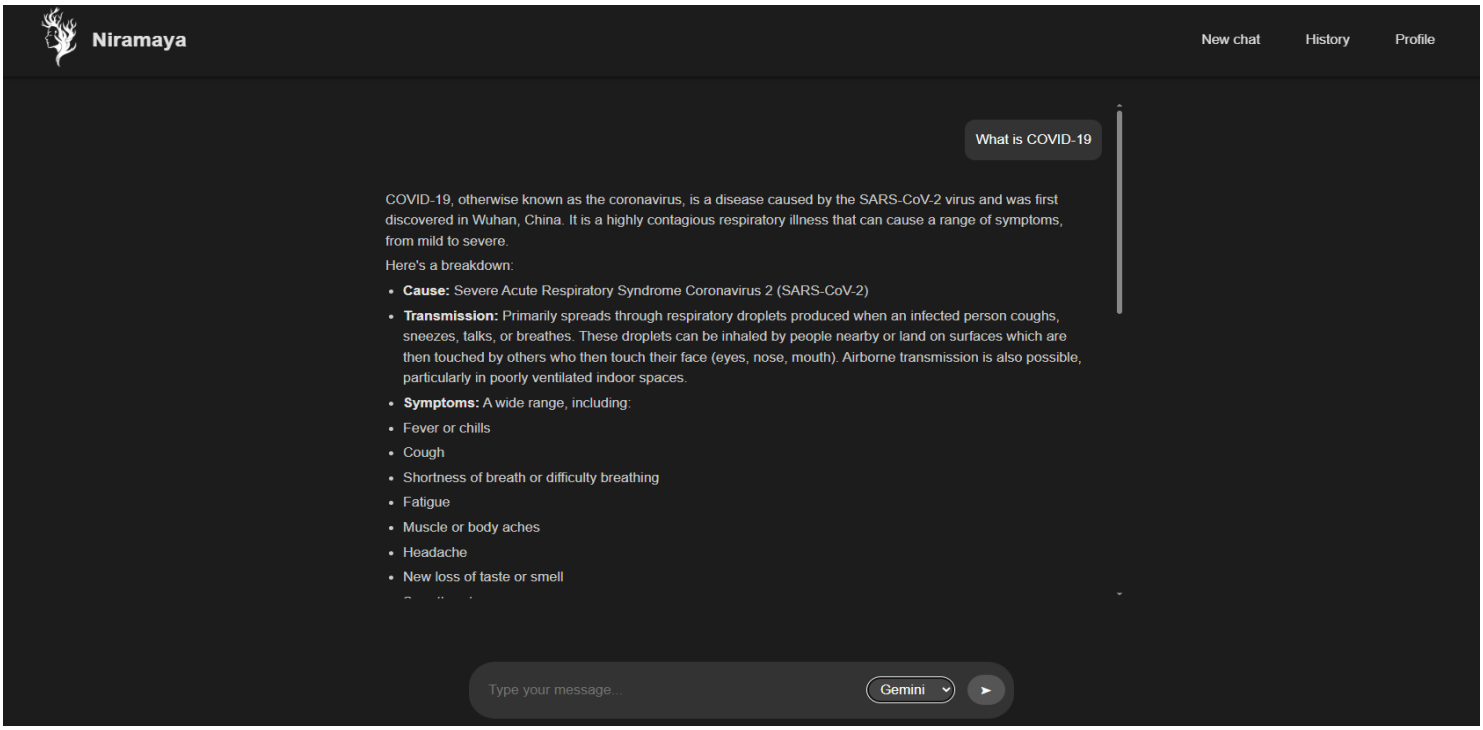
History



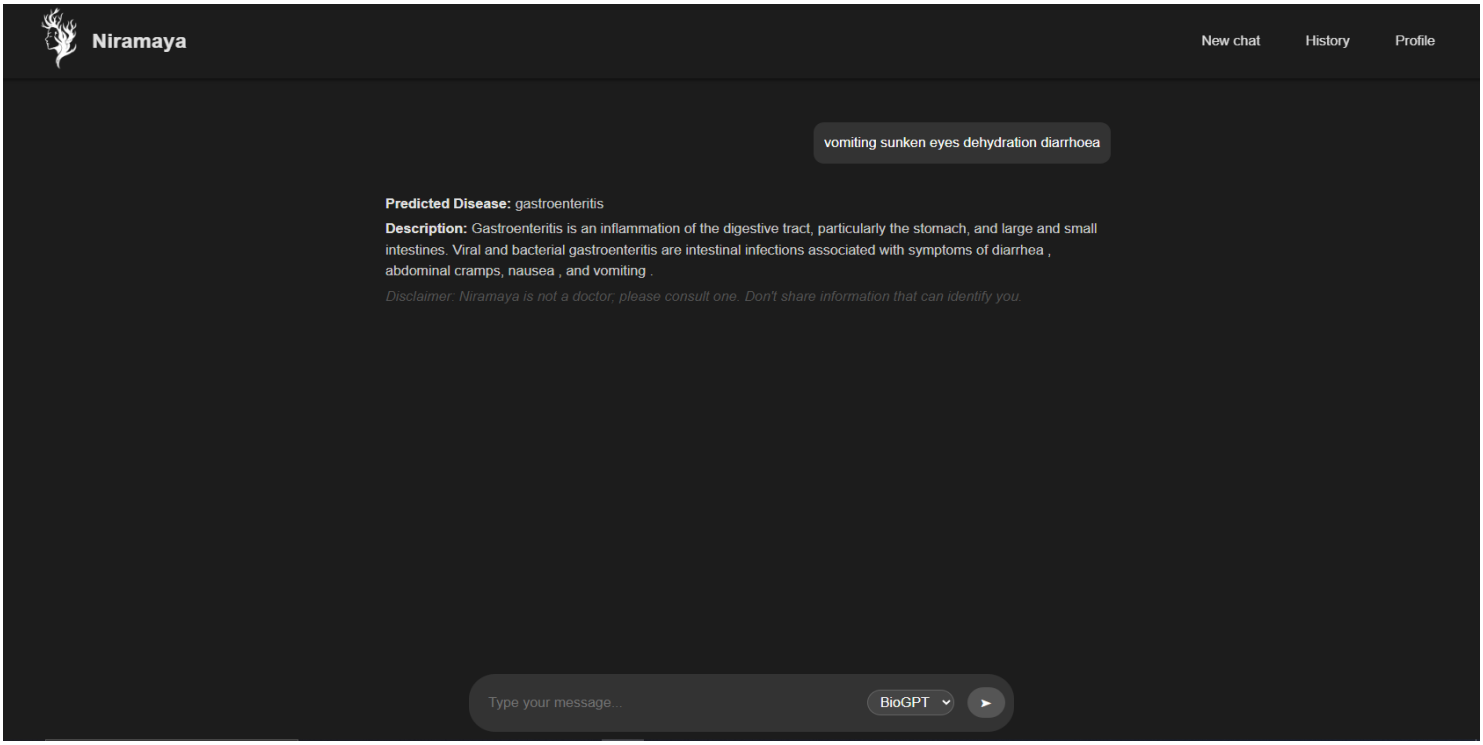
Response of BioGPT



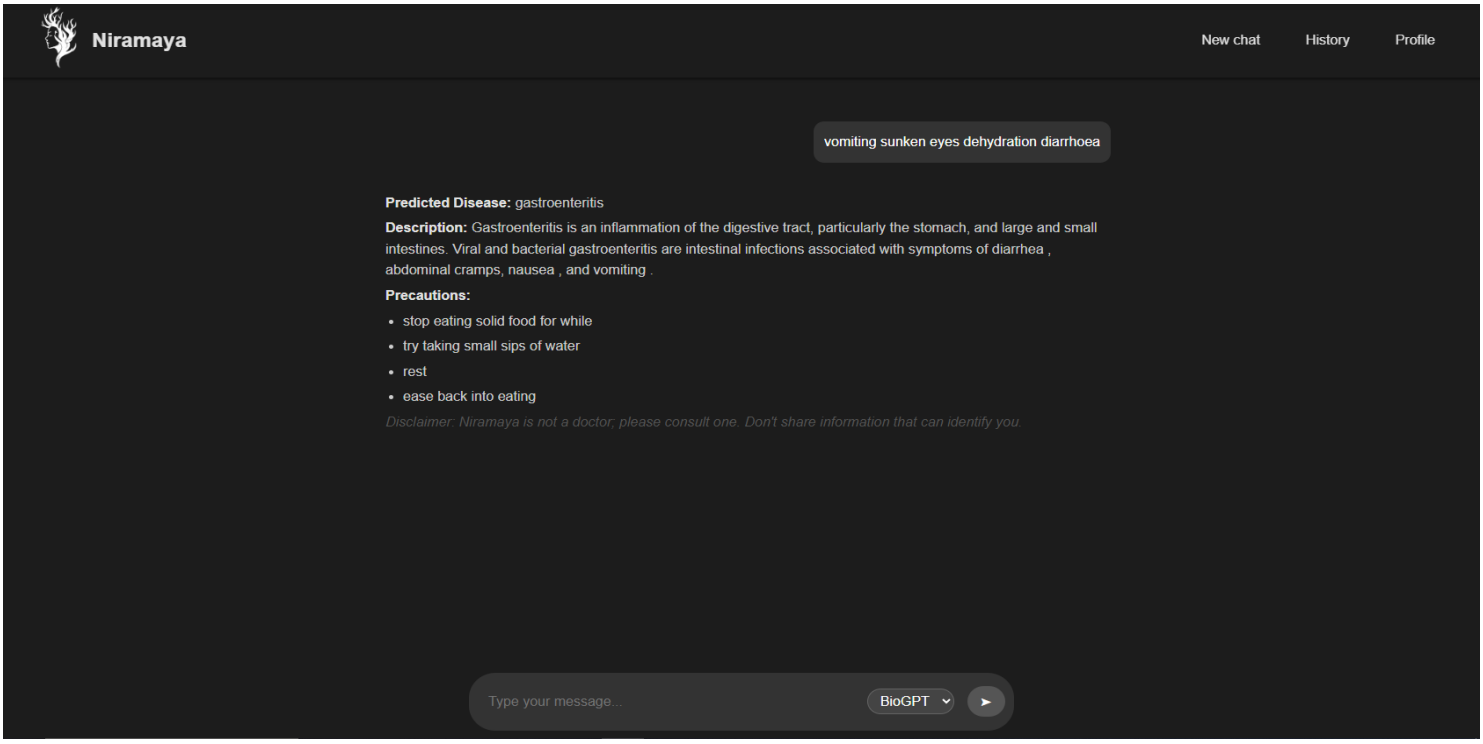
Response of Gemini



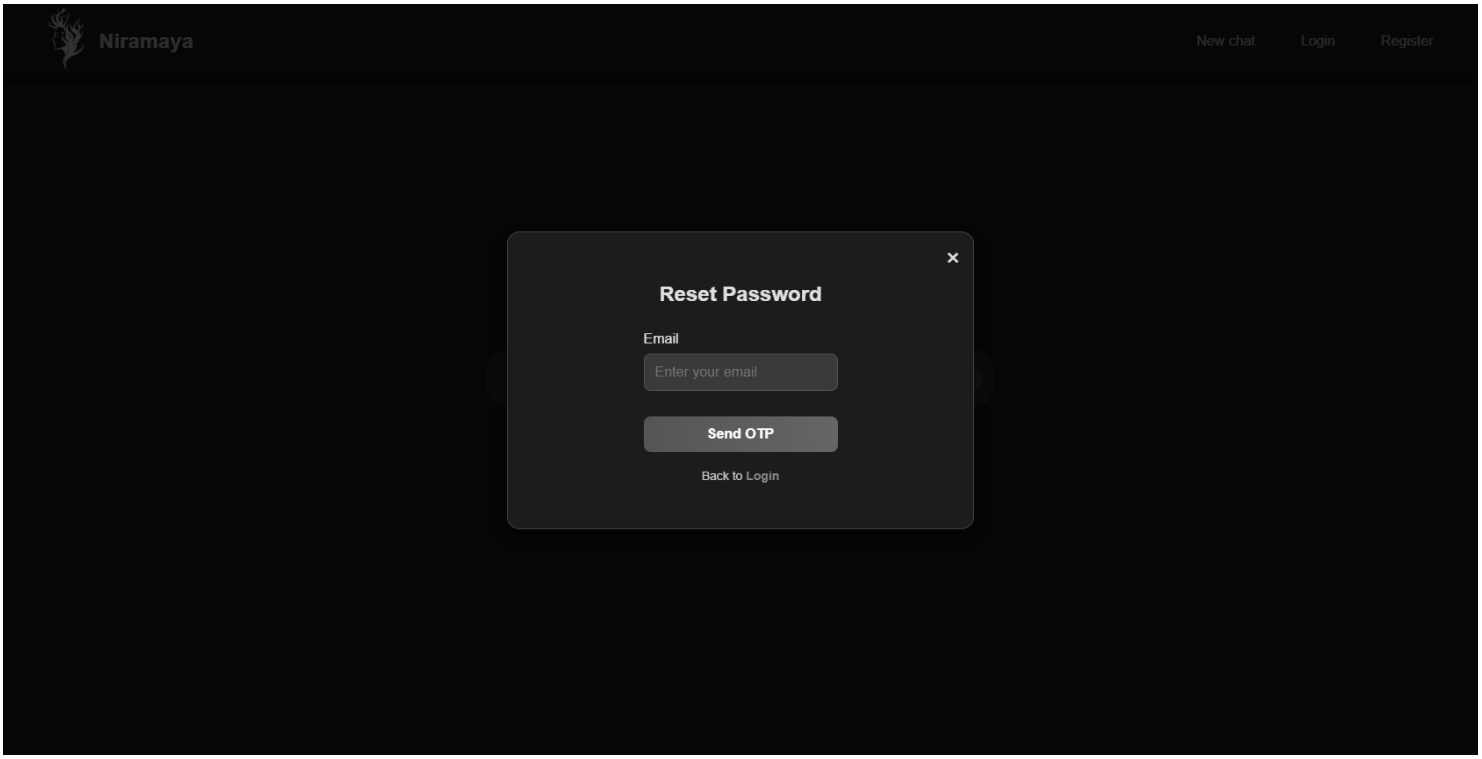
Response of LRv1

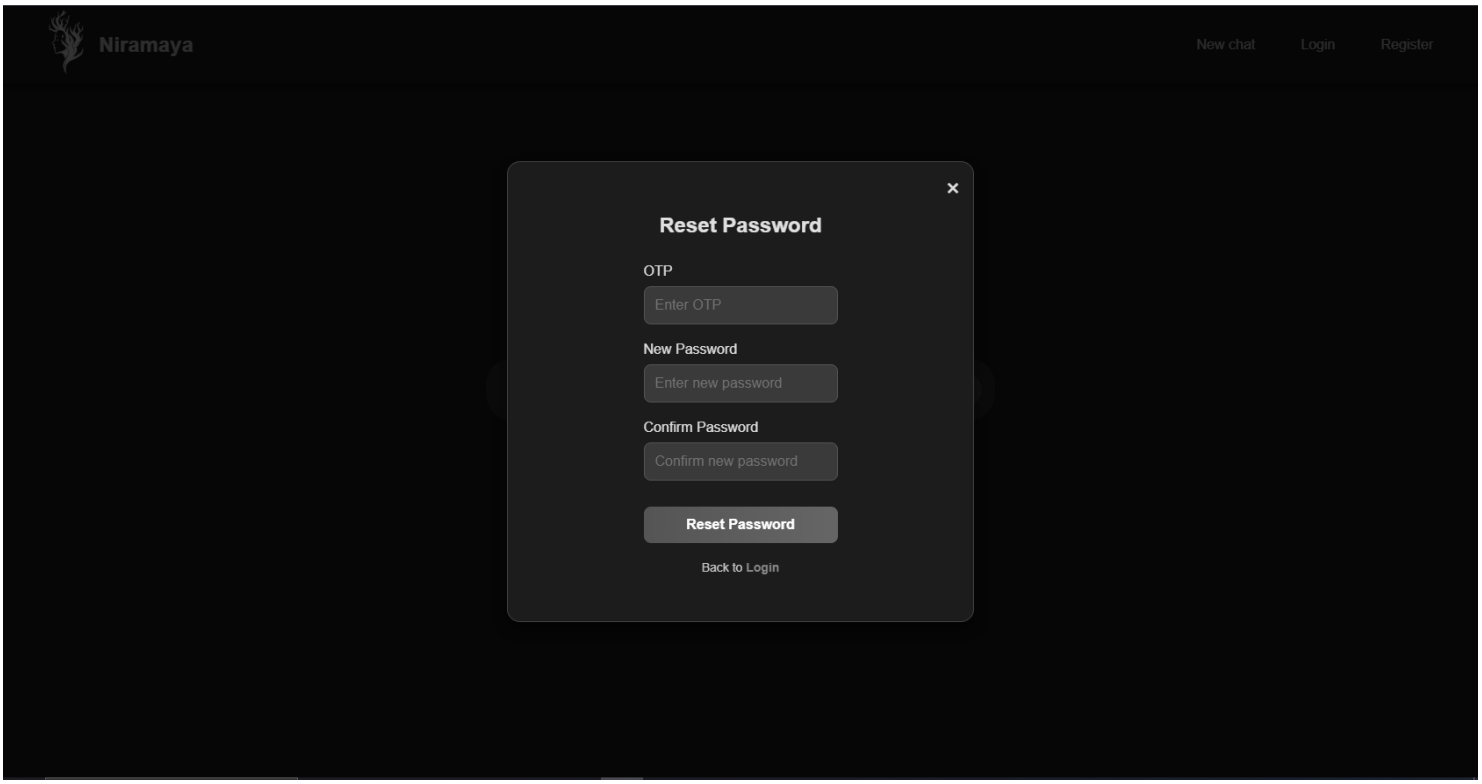


Response of LRv2

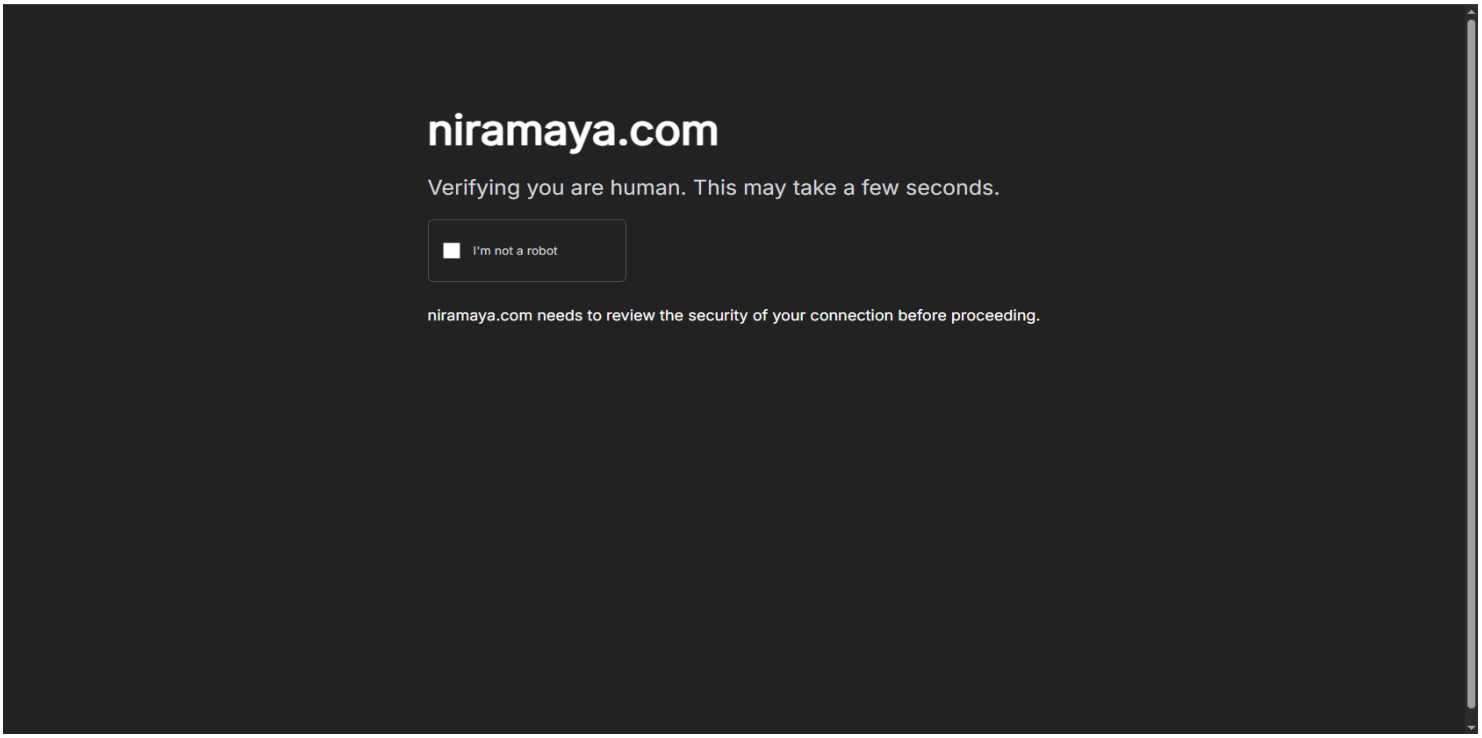


Reset Password

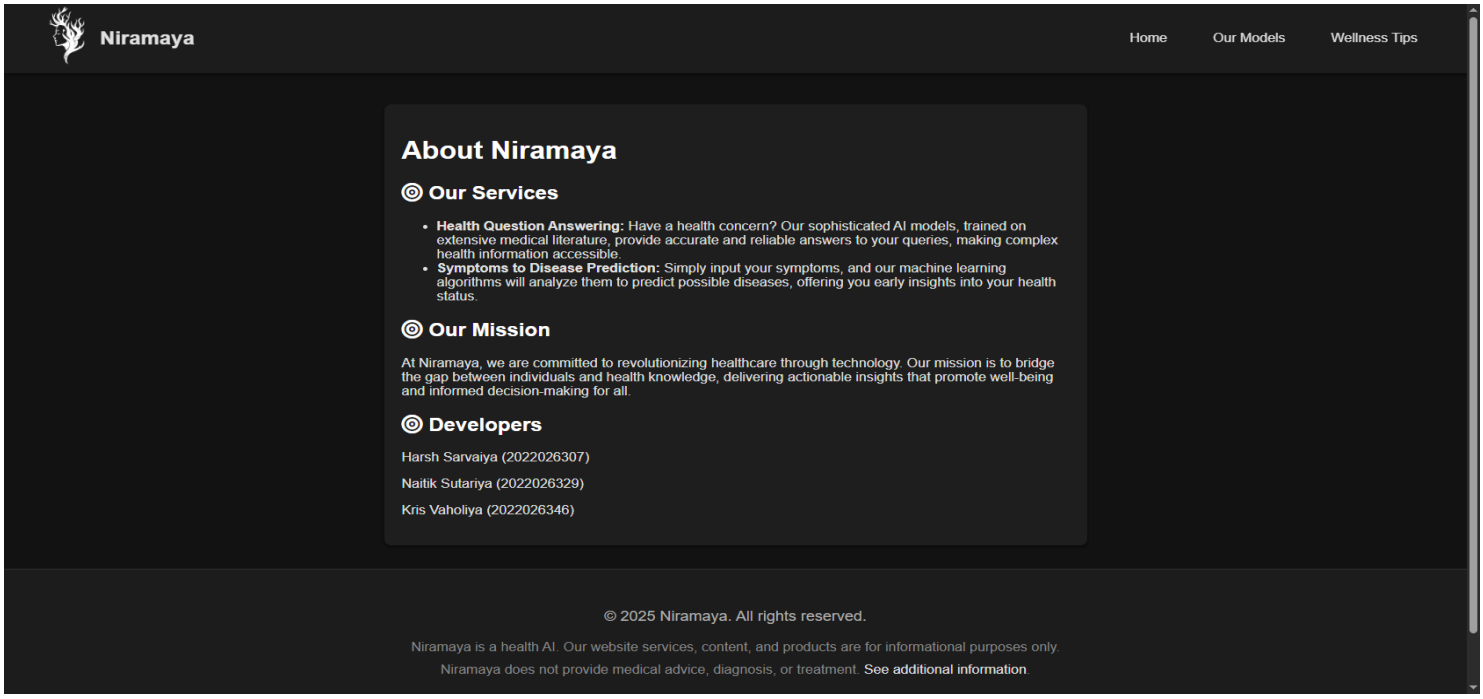




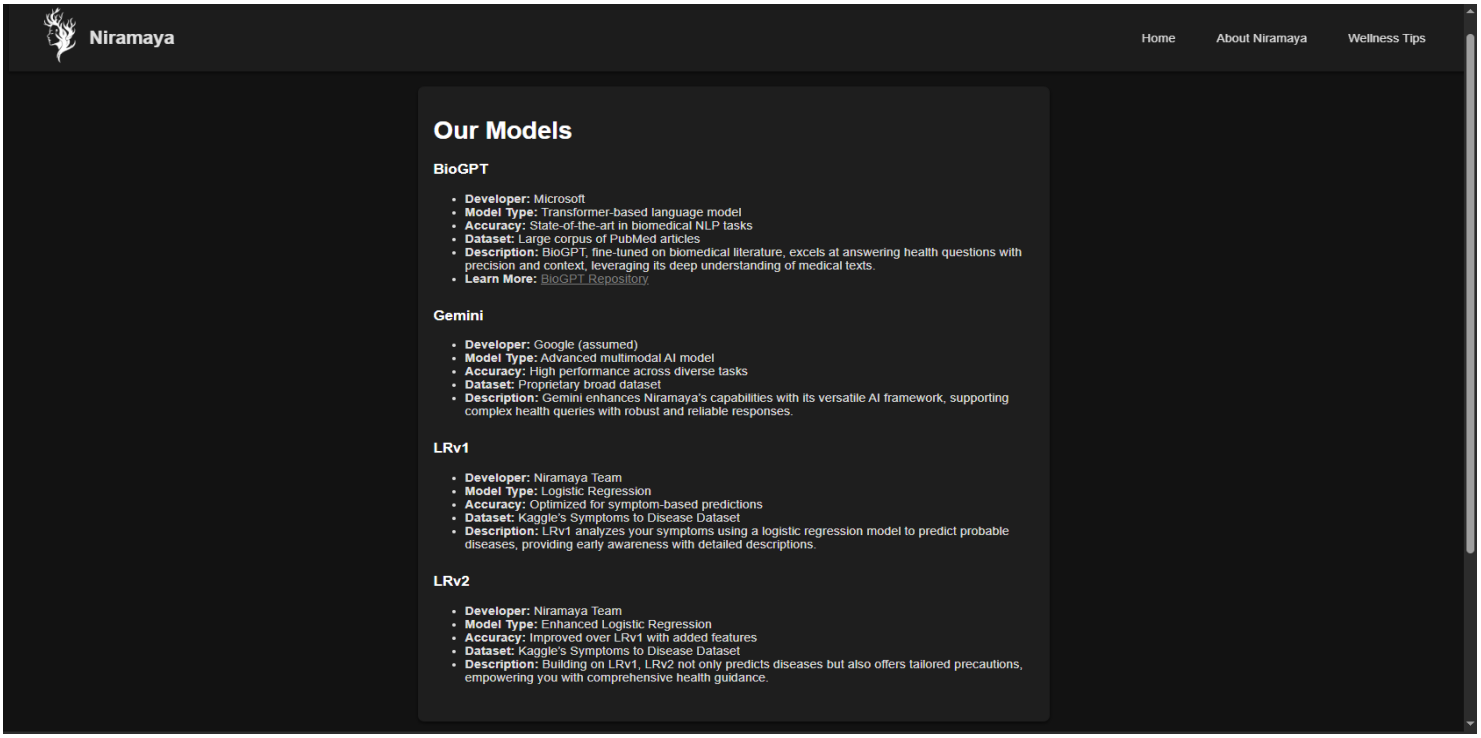
3. I’m Not a Robot Page



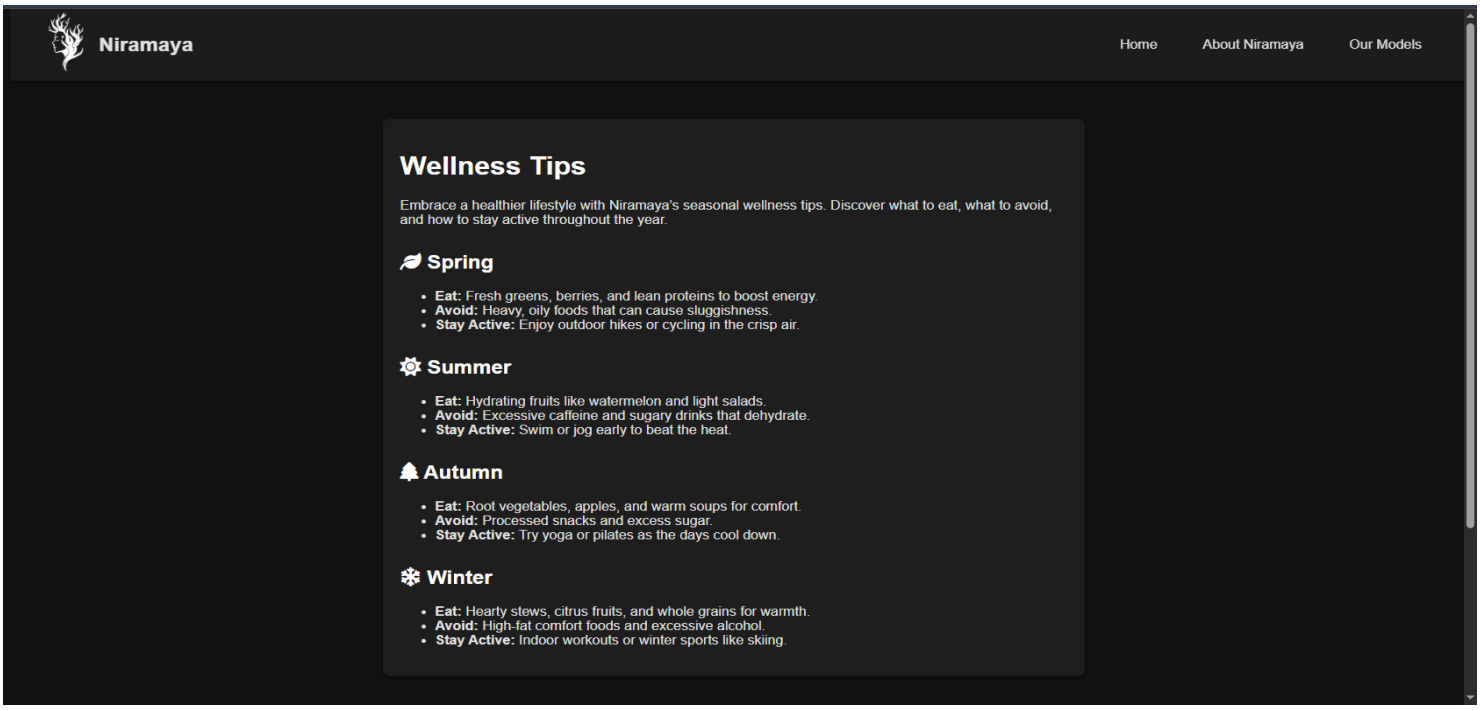
4. About Niramaya Page



5. Our Model Info Page



6. Wellness Tips Page



7. Testing

7.1 Testing Methodologies

To ensure the system's robustness and reliability, multiple testing methodologies were applied throughout the development cycle:

- **Unit Testing:**
  Individual components of the application were tested in isolation. This included testing functions related to user authentication, data validation, and the basic operations of AI model integration (e.g., sending a query to an ML model and retrieving a response). Unit testing ensures that each module performs as expected before combining them into a larger system.
- **Integration Testing:**
  Once individual modules passed unit tests, integration testing was conducted to verify that the different components (frontend, backend, AI models, and database) work together seamlessly. This testing phase focused on the flow of data between modules, such as ensuring that user inputs are correctly processed by the backend, passed to AI models, and the resulting predictions are accurately stored and retrieved from the database.
- **System Testing:**
  The complete application was tested as an integrated whole in an environment that closely resembled production. System testing involved end-to-end scenarios, such as a user logging in, inputting symptoms, receiving model predictions, and viewing the chat history. This comprehensive testing ensured that all features met the required functionality, performance benchmarks, and usability standards.
- **User Acceptance Testing (UAT):**
  Informal user acceptance testing was carried out by a group of target users to validate the overall user experience, interface design, and the relevance of the AI-generated outputs. Their feedback helped to refine the application's ease of use and ensure that it met real-world needs.

7.2 Test Cases

Test Cases	Status
User Login	Pass
User Registration	Pass
Member Addition	Pass
Member Remove	Pass
Health Question Query	Pass
Symptoms Checking	Pass
Check History	Pass

8. AI Model Summary

This section provides an overview of the AI model developed for symptom-to-disease prediction in the project. The model is central to the system's diagnostic functionality and operates as part of the backend through an integrated Flask API.

8.1 Model Type and Dataset

The trained model used is a **Logistic Regression** classifier, chosen for its efficiency and effectiveness in handling multi-class classification problems. The dataset used for training was sourced from **Kaggle**, combining data from the **National Health Portal (NHP)** and **Wikipedia**. It comprises a set of diseases (as labels) and corresponding symptoms (as features).

- **Model Type:** Logistic Regression
- **Dataset Source:** Kaggle (scraped from NHP and Wikipedia)
- **Number of Features:** All symptoms were one-hot encoded
- **Preprocessing:** Dataset was loaded from CSV and directly split into features (X) and labels (Y)

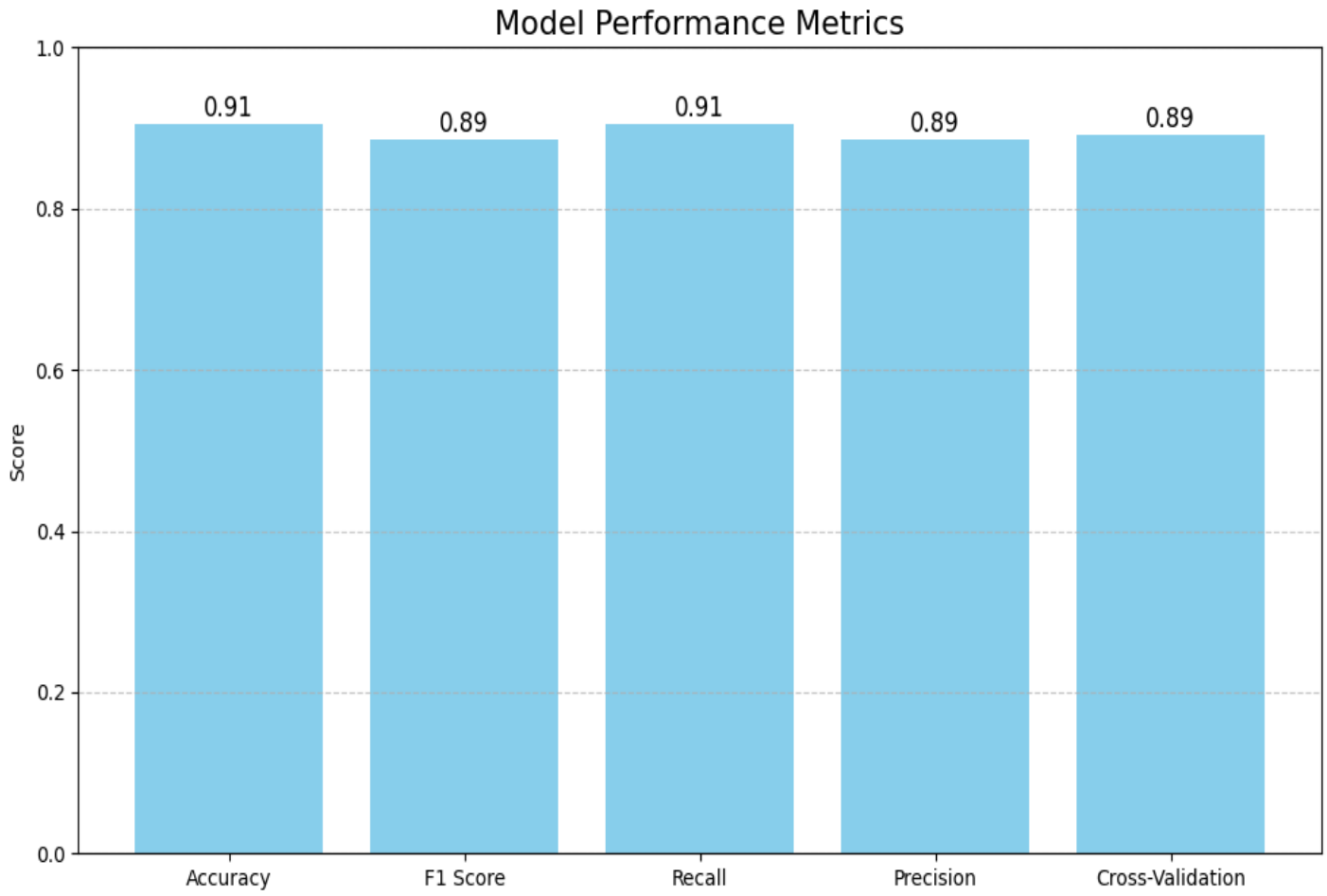
8.2 Training Procedure

The dataset was split using a **90:10 training-test ratio**. The model was trained using the full set of binary-encoded symptom features. A **5-fold cross-validation** technique was also applied to evaluate the generalizability of the model.

- **Training/Test Split:** 90% training, 10% testing
- **Cross-Validation:** 5-fold
- **Model Training Library:** scikit-learn

8.3 Model Evaluation

The model was evaluated using standard classification metrics. The results indicated a strong performance across all metrics, making it suitable for real-world symptom-based diagnosis tasks.



- **Accuracy:** 91%
- **Precision:** 89%
- **Recall:** 91%
- **F1 Score:** 89%
- **Cross-Validation Accuracy:** 89%

Note:

- *Accuracy* = (TP + TN) / (TP + FP + TN + FN)
- *Precision* = TP / (TP + FP)

- *Recall* = TP / (TP + FN)
- *F1 Score* = 2 × (Precision × Recall) / (Precision + Recall)
  Where TP = True Positives, FP = False Positives, TN = True Negatives, FN = False Negatives

8.4 Model Deployment and Integration

The trained model was saved using the **Joblib** library in .pkl format and integrated into the system via a **Flask API**. Additionally, a list of symptom features was stored separately (symptom_features.pkl) to ensure consistency in feature encoding during real-time inference.

- **Saved Model File:** logistic_regression_model.pkl
- **Feature Encoder File:** symptom_features.pkl
- **Integration Method:** Flask API
- **Inference Process:**
  1. Symptoms are input via the frontend
  2. Backend sets corresponding binary feature values
  3. Model returns predicted disease

9. Future enhancement

To improve the performance and usability of *Niramaya - Health-AI*, several enhancements are planned for future versions of the system:

1. **Improving Model Accuracy**

The current machine learning models (LRv1 and LRv2) can be trained on larger and more diverse datasets to improve the accuracy of disease prediction and precaution suggestions.

2. **Adding More Symptoms and Diseases**

Expanding the model to include a wider range of symptoms and diseases will allow the system to handle more complex or specific health conditions.

3. **User Feedback System**

Implementing a simple feedback option after each response will help collect user insights on the accuracy and helpfulness of predictions, which can be used to fine-tune the system.

4. **Multilingual Support (Basic)**

Adding support for a few regional languages will help non-English speaking users interact more easily with the system.

5. **User Interface Improvements**

Enhancing the frontend design for better navigation and responsiveness across devices (especially mobiles and tablets) will improve the user experience.

6. **Reminder Feature**

A basic reminder system can be added to notify users about regular health check-ins or to revisit symptoms after a few days.

10. References

1. **Scikit-learn Documentation**
   https://scikit-learn.org
   (Used for implementing machine learning models LRv1 and LRv2)

2. **Hugging Face Transformers**
   https://huggingface.co/transformers/
   (Used for integrating BioGPT model for question-answering features)

3. **Flask - Python Web Framework**
   https://flask.palletsprojects.com/
   (Backend framework used for API development)

4. **PostgreSQL Official Documentation**
   https://www.postgresql.org/docs/
   (Used for managing structured data like users, members, and chat history)

5. **pgAdmin**
   https://www.pgadmin.org/
   (Tool used to manage the PostgreSQL database in GUI format)

6. **Visual Studio Code (VS Code)**
   https://code.visualstudio.com/
   (Code editor used for frontend and backend development)

7. **Postman API Platform**
   https://www.postman.com/
   (Used for testing REST APIs during development)

8. **Kaggle**
   https://www.kaggle.com/
   (Used for sourcing the symptom-to-disease dataset for model training)

9. **Draw.io (diagrams.net)**
   https://draw.io/
   (Used to create ER diagrams and UML diagrams)

10. **dbdiagram.io**
    https://dbdiagram.io/
    (Used for generating visual ER diagrams from schema code)

11. **YouTube Tutorials and Open Source Projects**
    (Used as references for Flask, ML, and UI integration techniques)