Chatbot to known Individual Prakriti (Phenotype)

1. SRS Overview

This clause defines the normative content of the Software Requirements Specification (SRS) for the **Prakriti-Based Chatbot System**. The SRS is intended to serve as a reference for developers, testers, project managers, and stakeholders to understand the software's functionality, features, and non-functional requirements. The structure and content of this document are aligned with the project's information management and quality assurance policies.

The Prakriti-Based Chatbot is designed to leverage traditional Ayurvedic knowledge and modern machine learning techniques to identify the user's **Ayurvedic Dosha (Prakriti)** and deliver **personalized healthcare recommendations**.

2. Purpose

The purpose of the software is to provide an interactive, AI-powered chatbot that helps users determine their **Prakriti (Vata, Pitta, or Kapha)** by analyzing responses to a predefined set of questions. The chatbot uses a **Naive Bayes classifier** to predict the user's Dosha and recommends personalized Ayurvedic advice on **diet, lifestyle, yoga, and seasonal care**. It aims to promote holistic wellness by integrating ancient wisdom with modern technology and providing real-time, accessible healthcare support.

3. Scope

a) Software Product Name

- Prakriti-Based Chatbot System

b) What the Software Will Do

- Interact with users through a dynamic chatbot interface
- Ask a structured set of Ayurvedic questions
- Analyse responses to predict the user's dominant Dosha using a **Naive Bayes classification algorithm**
- Provide **personalized recommendations** based on the user's Prakriti
- Enable **administrative control** over users, questions, and recommendation data

c) Application and Benefits

The software will be applied in the Ayurveda-based wellness and healthcare domain. It is designed for individuals seeking customized Ayurvedic guidance without the need for in-person consultations.
Benefits and Objectives include:

- Promoting **self-awareness** of body constitution (Prakriti)
- Providing **accessible Ayurvedic consultation** using AI
- Encouraging **preventive healthcare** through lifestyle suggestions
- Facilitating **digitization of traditional medicine** systems
- Supporting researchers with Prakriti analytics and user data (admin side)

d) Consistency with Higher-Level Specifications

This software aligns with broader system-level goals of promoting digital healthcare, Ayurvedic integration into modern systems, and personalized wellness solutions as envisioned by national health initiatives and WHO's traditional medicine strategies.

4. Product Perspective

This chatbot system is a standalone software application but can also function as a modular element in a larger health guidance ecosystem or Ayurvedic diagnostic platform. The product primarily operates within a client-server model, where users interact with a web-based client (frontend), and the backend handles data processing, Prakriti prediction, and interfacing with other services.

4.1 System Interfaces

- Chatbot Web App ↔ Django Backend Server:
    - HTTP-based REST API.
    - Functionality includes data submission (user answers), Prakriti prediction, and result fetching.
- Backend Server ↔ Database:
    - CRUD operations using Django ORM with PostgreSQL.
    - Stores user data, response logs, admin inputs, dosha tips, etc.
- Backend ↔ ML Model:
    - Internal API or function call to make Naive Bayes-based predictions on user data.
- Admin Panel ↔ Server:
    - Allows secure login and editing of rules, dosha content, and user analytics.

4.2 User Interfaces

- Clean, mobile-responsive web UI using React.js and TailwindCSS.
- Features: Chat interface, visual Prakriti result, personalized health tips.
- UI designed with a user-friendly style guide (consistent buttons, forms, dark/light modes).
- Accessible design for all age groups.

4.3 Hardware Interfaces

- Web application accessible on devices with modern web browsers (PCs, smartphones, tablets).
- No specialized hardware required.

- Supports basic input devices (keyboard, touch input, microphone if extended to voice).

4.4 Software Interfaces

- Operating System: Cross-platform (Windows/Linux/Mac for admin, Android/iOS for users).
- Database: PostgreSQL
    - Mnemonic: postgres
    - Version: ≥13.x
    - Source: PostgreSQL.org
- Backend: Django 4.x
- Frontend: React.js 18+, TailwindCSS
- ML Model: Naive Bayes classifier using scikit-learn
- Libraries/APIs: Axios for API calls, D3.js for graphs (if needed for visual Prakriti results).

4.5 Communications Interfaces

- Protocols: HTTPS for secure communication.
- APIs: RESTful APIs between frontend and backend.
- Optional Future Integration: Firebase Cloud Messaging (FCM) for real-time updates or notifications.

4.6 Memory Constraints

- Requires 2GB RAM minimum for hosting backend and ML model.
- PostgreSQL with optimized indexes ensures storage efficiency.
- Frontend loads are optimized using lazy loading and component-level caching.

4.7 Operations

- **Modes**: Interactive (chat-based), Admin (data editing, monitoring), Passive (display-only for users).
- **Unattended Operations**: Background ML model training or logging does not require user interaction.
- **Support Functions**:
    - Data analytics for admin.
    - Auto-save and error logging.
- **Backup**:

- Nightly database dump.
- Admin-exportable logs and configuration.

4.8 Site Adaptation Requirements

- Adaptable for:
  - Different languages (via i18n support).
  - Regional Ayurvedic content (editable by admin).
  - Seasonal and location-based Prakriti advice (future enhancement).
- Initialization sequence includes uploading:
  - Default dosha advice,
  - Admin credentials,
  - Questionnaire.

4.9 Interfaces with Services

- Currently standalone, but can integrate with:
  - SaaS: Firebase (for hosting/notifications), AWS (for cloud deployment).
  - APIs: Text summarization, weather APIs (planned) for context-aware Ayurvedic advice.
  - Google Forms/Sheets: For collecting user feedback in initial beta release.

5. Product Functions

The Prakriti-based Chatbot Web Application is designed to perform the following major functions:

5.1. User Interaction and Data Collection

- Facilitate user registration and login.
- Present a series of predefined Prakriti assessment questions to users.
- Collect and store user responses securely.
- Allow users to update their profiles and view past results.

5.2. Prakriti Analysis and Prediction

- Use the Naive Bayes algorithm to predict the user's Tridosha type (Vata, Pitta, Kapha).
- Store analysis results in the backend database for personalized guidance.

- Visualize Prakriti results using charts and textual descriptions.

5.3. Personalized Ayurvedic Recommendations

- Provide customized dietary, lifestyle, and yoga recommendations based on the predicted Prakriti type.
- Display seasonal tips and preventive suggestions for each Dosha.
- Allow users to view historical insights for Dosha balance monitoring.

5.4. Admin Control and Management

- Admin authentication and dashboard access.
- Manage users' data, questions, and Prakriti mappings.
- Update the database with new health guidelines, Prakriti-specific content, and seasonal recommendations.

5.5. Educational Content Delivery

- Serve informative content regarding Ayurveda, Tridosha theory, lifestyle tips, and dosha balancing strategies.
- Provide infographics, PDFs, and text-based learning modules for user awareness and self-learning.

5.6. Notification and Reminders System

- Send periodic reminders for wellness activities based on user Prakriti (e.g., daily yoga or diet alerts).
- Notify users about seasonal changes and suggest relevant adaptations.

5.7. Data Logging and Analytics

- Record user responses, predictions, and interactions for tracking health patterns.
- Enable admin to access anonymized analytics to improve system learning and content relevance.

5.8. Accessibility and Interface Support

- Provide a responsive user interface for desktop and mobile users.
- Ensure accessibility for different age groups using a simple conversational UI with chatbot interactions.

6. User Characteristics

The Prakriti-based Chatbot Web Application is designed for a diverse group of users with varying levels of familiarity with Ayurveda and digital technologies. Understanding these

user characteristics is essential for ensuring usability, accessibility, and effective health guidance.

6.1. General User Categories

- General Public / Health-Conscious Individuals
    - Age Range: 18–60+
    - Interest in natural wellness, preventive healthcare, and Ayurveda.
    - Educational Level: Ranges from high school graduates to postgraduates.
    - Technical Expertise: Basic to moderate computer/mobile usage skills.
- Ayurveda Enthusiasts / Students
    - Higher awareness and understanding of Prakriti and Dosha concepts.
    - Likely to explore more advanced features and content within the application.
- Health Coaches and Wellness Experts
    - Interested in using the system for preliminary assessments and guidance.
    - May provide feedback or expect in-depth analysis to aid clients.
- Administrators / System Supervisors
    - Moderate to high technical expertise.
    - Responsible for content management, user analytics, and system updates.

6.2. Characteristics Influencing Usability

- Varying Educational Backgrounds
    - The interface must use simple, easy-to-understand language and offer tooltips or visuals to explain Ayurvedic terms.
    - Multilingual support may be considered in the future.
- Digital Literacy
    - Most users are expected to be familiar with basic digital tools like browsers and mobile apps.
    - The interface should avoid complex navigation and use chatbot-style interactions for ease.
- Accessibility Needs
    - Users may include individuals with visual impairments or age-related limitations.
    - The design must be responsive, with high-contrast themes, adjustable font sizes, and screen-reader compatibility.

- Time-Conscious Behavior
    - Many users prefer quick results and brief interactions.
    - Hence, the questionnaire and recommendation system must be streamlined for completion within 3–5 minutes.

6.3 Implications for Requirement Design

These characteristics justify several later requirements such as:

- A simplified conversational UI (chatbot-style) for user interaction.
- Tooltips, infographics, and glossary links to explain Ayurvedic concepts.
- Minimalist layout with mobile-first design for broader accessibility.
- Customizable alerts and reminders catering to different user schedules and preferences.
- Accessibility features to support inclusive use.

7. Limitations

The development and deployment of the Prakriti-based Chatbot Web Application are subject to the following constraints and limitations, which influence system architecture, implementation, and operation.

a) Regulatory Requirements and Policies

- The system must comply with data privacy regulations such as India's IT Act and any applicable healthcare data protection norms (e.g., HIPAA if expanded internationally).
- Ayurvedic advice provided must align with guidelines issued by AYUSH (Ministry of Ayurveda, India) and avoid diagnosing or treating medical conditions.

b) Hardware Limitations

- Optimized for use on mobile devices and desktops with basic hardware specifications.
- Does not require special hardware acceleration, sensors, or wearables for primary functionality.
- Assumes internet connectivity for real-time processing.

c) Interfaces to Other Applications

- Currently, there is no integration with external EHR (Electronic Health Records) or medical systems.
- Future scope includes optional integration with weather APIs or fitness applications for environment-aware suggestions.

d) Parallel Operation

- The system is designed for single-user interaction at a time (chatbot-based). Multi-session handling is required only on the server-side to accommodate concurrent users.

e) Audit Functions

- Basic logging of user interactions for session continuity and user analytics.
- No built-in forensic auditing or deep activity tracking unless authorized by the administrator.

f) Control Functions

- Admin access is limited to managing user database, updating content, and monitoring logs.
- No system-level override functions are provided to users.

g) Higher-Order Language Requirements

- The primary version of the application will use English.
- Future updates may include support for Hindi and other regional languages through internationalization (i18n).

h) Signal Handshake Protocols

- Not applicable for web-based systems relying on HTTP/HTTPS protocol.
- WebSocket or AJAX used for real-time interaction does not require traditional signal handshakes like XON-XOFF.

i) Quality Requirements (e.g., Reliability)

- The application must maintain 99% uptime, excluding scheduled maintenance.
- The chatbot must produce consistent outputs for identical inputs to ensure predictability in health suggestions.

j) Criticality of the Application

- The chatbot is classified as **non-critical** as it offers **wellness and lifestyle guidance**, not medical diagnosis or emergency response.
- User disclaimer acceptance is mandatory before usage.

k) Safety and Security Considerations

- Secure user authentication for admin access.
- Data stored is anonymized and encrypted to prevent misuse.
- No sensitive medical data or personal identifiers are stored without user consent.

l) Physical/Mental Considerations

- Designed for low cognitive load: uses simple navigation, calming colors, and concise question formats.
- Meant to be usable by people with average digital skills and no medical background.

m) Limitations from Other Systems

- The chatbot uses rule-based plus probabilistic analysis (Naive Bayes), which may have reduced accuracy in the absence of consistent user input.
- Seasonal/environmental recommendations rely on third-party data sources (e.g., weather APIs), which may cause latency or inconsistency.

8. Assumptions and Dependencies

The following assumptions and dependencies have been made while defining the requirements for the Prakriti-based Chatbot Web Application. Any changes to these factors may impact the software's requirements, functionality, or performance.

8.1 Assumptions

- **Web-Based Access**
  It is assumed that users will access the system via modern web browsers such as Chrome, Firefox, or Microsoft Edge.
  ➤ *Browser compatibility is essential for proper functionality.*
- **Availability of Hosting Environment**
  The application will be hosted on a stable cloud or institutional server with reliable internet connectivity and support for LAMP/MEAN stack environments**.**
- **User Internet Connectivity**
  The system assumes that users have a stable internet connection to enable real-time chatbot interactions.
- **Admin Expertise**
  The system assumes that the admin possesses basic technical skills required to manage content, update datasets, and monitor system logs via the admin dashboard**.**
- **End-User Literacy and Device Access**
  Users are expected to have basic literacy in the supported language (initially English) and access to an internet-enabled smartphone, tablet, or computer**.**
- **No Legal Restrictions on Wellness Tools**
  It is assumed that there are no legal restrictions in the target regions on providing Ayurvedic wellness suggestions through digital platforms.

8.2 Dependencies

- **Operating System Support**
  The application is designed for deployment on Linux or Windows-based servers compatible with Python and Django.
  ➤ *Changes in OS environments may require backend adaptation.*

- **External Libraries and APIs**
  The system depends on third-party Python libraries such as scikit-learn, NLTK, and Django, along with optional weather APIs for dynamic recommendations.
  ➤ *Deprecation or changes in these libraries/APIs may affect system behaviour.*
- **Ayurvedic Knowledge Base**
  The chatbot's Prakriti prediction and health advice rely on a validated and expert-reviewed Ayurvedic dataset.
  ➤ *Any expansion or revision in Ayurvedic content will require logic updates.*
- **Browser Storage and Cookies**
  The application may use browser storage (e.g., localStorage or cookies) to store session-related data.
  ➤ *Disabling storage mechanisms may affect session continuity or personalization.*

8.3 Requirement Allocation Table

Requirement	Software Element(s)	Status
User login, registration, and authentication	Frontend, Backend (Django Auth), Database	Version 1 (Implemented)
Prakriti questionnaire form and data collection	Chatbot Frontend UI, Backend Logic	Version 1 (Implemented)
Prakriti prediction using Naive Bayes	Machine Learning Module (scikit-learn), Backend	Version 1 (Implemented)
Personalized Ayurvedic recommendations	Backend, Ayurvedic Knowledge Base	Version 1 (Implemented)
Admin dashboard for content and dataset management	Admin UI, Backend	Version 1 (Implemented)
Dynamic seasonal tips based on location	Weather API Integration, Backend	Version 2 (Planned)
Multilingual support (e.g., Hindi, Regional Languages)	Frontend, i18n Module	Version 2 (Planned)
User session management and local storage	Frontend (Browser Storage), Backend Session Handler	Version 1 (Implemented)
Analytics for admin (user trends, Prakriti distribution)	Backend Analytics Module, Admin Dashboard	Version 2 (Planned)
Notifications and reminders (e.g., health tips, check-ins)	Notification Service (e.g., FCM), Backend Scheduler	Version 3 (Planned)
Accessibility features (text-to-speech, high contrast UI)	Frontend UI Enhancements	Version 3 (Planned)

Notes

- Some requirements such as *weather-based health tips*, *analytics*, and *notifications* have been deferred to **future versions** due to integration complexity and testing needs.
- Requirements spanning multiple software elements (e.g., Prakriti prediction logic) will require coordinated implementation and testing across the frontend, backend, and machine learning modules.
- Dependencies like external APIs and Ayurvedic content updates may necessitate re-apportioning in future updates.

8.4 Non-Functional Requirements

ID	Requirement
NFR-01	The system shall be compatible with major modern browsers (Chrome, Firefox, Edge).
NFR-02	The system shall support responsive design for mobile and desktop platforms.
NFR-03	The system shall respond to user input within 2 seconds under normal load.
NFR-04	All user data shall be stored securely using encrypted connections (HTTPS).
NFR-05	The system shall support up to 100 concurrent users without performance degradation.
NFR-06	The system shall be modular to support future enhancements like voice input or multilingual support.

11. External Interfaces

The system interacts with several external components to ensure seamless chatbot functionality and data flow. Each interface is described below:

11.1. User Interface (Web-Based Frontend)

- Name: User Interaction Interface
- Purpose: To allow users to interact with the chatbot via questions and receive personalized Prakriti analysis and suggestions.

- Source/Destination: End-user (browser)
- Valid Range / Accuracy: Accepts standard text input; no special characters expected in responses
- Units of Measure: UTF-8 encoded text
- Timing: Real-time, synchronous interaction with prompt responses
- Relationships: Triggers backend chatbot logic upon message submission
- Data Formats: JSON for AJAX-based communication; HTML/CSS for rendering
- Command Formats: User inputs are sent via HTTP POST requests
- Data Items: User responses to questions, chatbot replies, session IDs

11.2. Admin Dashboard Interface

- Name: Admin Control Panel
- Purpose: Allows admin to manage user data, review logs, update question sets or Ayurvedic content
- Source/Destination: Admin (via browser)
- Valid Range / Accuracy: Form validation for data consistency
- Units of Measure: UTF-8 encoded strings and JSON payloads
- Timing: Asynchronous updates and CRUD operations
- Relationships: Writes to/reads from backend database
- Data Formats: JSON (backend communication), HTML/CSS
- Command Formats: HTTP POST/GET/PUT requests
- Data Items: Question bank, Ayurvedic dataset, user activity logs

11.3. Backend Processing Engine

- Name: Django Backend API
- Purpose: Processes user inputs, performs Prakriti analysis, generates recommendations
- Source/Destination: User Interface and Admin Interface
- Valid Range / Accuracy: Validated user inputs; trained Naive Bayes model for prediction
- Units of Measure: UTF-8 text, numeric scores
- Timing: Real-time processing with low latency

- Relationships: Interfaces with database and external Python libraries
- Data Formats: JSON input/output, form data
- Command Formats: Function calls and REST API endpoints
- Data Items: User answers, dosha classification results, recommendations

11.4. Database Interface

- **Name:** PostgreSQL / SQLite Database
- **Purpose:** Stores user responses, results, admin logs, and static Ayurvedic content
- **Source/Destination:** Backend API
- **Valid Range / Accuracy:** Ensured via schema constraints
- **Units of Measure:** Structured data (strings, integers, timestamps)
- **Timing:** CRUD operations triggered on user/admin actions
- **Relationships:** Central store for all modules
- **Data Formats:** Relational data in SQL format
- **Command Formats:** SQL queries
- **Data Items:** User table, Prakriti results, question sets, session data

12. Functions

The fundamental actions performed by the Prakriti-Based Chatbot Web Application are described below:

a) Validity Checks on the Inputs

- All user inputs (responses to questions) are validated to ensure:
    - Only expected answer formats (text-based, predefined options) are accepted.
    - No malicious input (such as SQL injection or script injection) is passed forward.
    - Mandatory fields are filled; empty responses trigger prompts for completion.

---

b) Exact Sequence of Operations

1. User accesses the web application via browser.
2. System initializes a new user session.
3. Chatbot presents a series of Prakriti assessment questions sequentially.

4. User responds to each question.
5. After all responses are collected:
    - Preprocessing is performed (e.g., cleaning text inputs, converting to numerical form if needed).
    - The Naive Bayes model processes the inputs to classify into Vata, Pitta, or Kapha Prakriti.
6. Based on the predicted Prakriti:
    - Personalized suggestions (food, lifestyle, yoga tips) are generated.
7. Results and recommendations are displayed to the user.
8. User can optionally download or email the report.
9. The session details are stored in the database for analytics/admin review.

c) Responses to Abnormal Situations

1) Overflow

- Application ensures input buffers are properly sized to avoid buffer overflow.
- Inputs exceeding expected limits are truncated and flagged.

2) Communication Facilities

- If there is a communication failure (e.g., loss of server connection):
    - A "Connection Lost" message is displayed.
    - The application retries to reconnect automatically for a set number of attempts.

3) Hardware Faults and Failures

- In case of server downtime or crash:
    - A server maintenance error page is shown.
    - Automatic system health checkers notify the admin.

4) Error Handling and Recovery

- Client-side and server-side validations ensure smooth operations.
- Exceptions (like database write failure, API call failure) are caught and logged.
- User-friendly error messages are displayed instead of system errors.

d) Effect of Parameters

- Input Parameters: User's answers heavily influence the prediction of Prakriti type.
- Session Parameters: Session ID maintains continuity across interactions.

- Model Parameters**:** The Naive Bayes model's hyperparameters impact classification accuracy (handled during system setup, not user-side).

e) Relationship of Outputs to Inputs

1) Input/Output Sequences

- Sequence:
    - Inputs (user answers) → Processing (Prakriti prediction) → Output (recommendations).
- Each input answer is mapped logically to corresponding features used by the prediction model.

2) Formulas for Input to Output Conversion

- User responses are encoded numerically or categorically.
- The Naive Bayes model uses probability calculations:

  $$P(\text{Dosha}|\text{Responses}) = P(\text{Responses}|\text{Dosha}) \times P(\text{Dosha}) / P(\text{Responses})$$

- The Dosha with the highest posterior probability is selected.
- Mapped recommendations are fetched based on the identified Dosha.

13. Usability Requirements

The Prakriti-Based Chatbot Web Application aims to deliver a highly usable, accessible, and user-friendly experience to both general users (seeking Prakriti analysis) and admin users (managing content and user interactions). Usability objectives have been defined with respect to the following measurable criteria:

1. Effectiveness

- Objective: Users should be able to complete the Prakriti assessment process with minimal errors or confusion.
- Measure:
    - ≥ 95% of users should complete the full assessment without needing help or support.
    - Error rate during input (e.g., invalid selections) should be less than 2%.

2. Efficiency

- Objective: The application should support rapid and smooth user interaction.
- Measure:

- Average time to complete the Prakriti analysis and view results should be ≤ 5 minutes.
- Response time per interaction (question answered → next response displayed) should be ≤ 1 second.
- Admin should be able to update datasets or user logs within 2-3 clicks.

3. Satisfaction

- Objective: Users should feel satisfied with the usability, visual design, and overall experience.
- Measure:
  - ≥ 90% positive feedback in user surveys regarding ease of use and clarity of chatbot interactions.
  - Net Promoter Score (NPS) ≥ +30 from user feedback.
  - UI should maintain consistency in fonts, button layouts, and colors to reduce cognitive load.

4. Avoidance of Harm

- Objective: Ensure that the system does not present health misinformation or confuse users.
- Measure:
  - Recommendations are clearly labeled as *"Ayurvedic Suggestions"*, not medical prescriptions.
  - Disclaimer shown before result screen stating: “This tool is for informational and wellness purposes only.”
  - Admin must validate and approve all content updates to avoid incorrect or culturally sensitive recommendations.

5. Accessibility

- Objective: Support usability for a diverse user base including users with basic digital literacy.
- Measure:
  - Interface should be operable via keyboard only.
  - Text contrast ratios must comply with WCAG 2.1 AA standards.
  - Language used should be simple, non-technical, and clear.

6. Learnability

- Objective: First-time users should be able to use the chatbot without training.
- Measure:
    - 90% of new users can reach the result page on their first try without external help.
    - Tooltips and placeholder texts guide user actions contextually.

14. Performance Requirements

14.1 Static Numerical Requirements:

- **a) The Number of Terminals to be Supported**:
    - The system should support at least 100 unique devices simultaneously (e.g., smartphones, desktops, tablets).
- **b) The Number of Simultaneous Users to be Supported**:
    - The system should support up to 1000 concurrent users without significant performance degradation.
- **c) The Amount and Type of Information to be Handled**:
    - The system should process text-based input of up to 500 characters per user interaction.
    - The application should handle up to 50,000 records of user responses and recommendations in the database.

14.2 Dynamic Numerical Requirements:

- **a) Number of Transactions and Tasks**:
    - The system should handle at least 100 transactions per minute (e.g., user query submissions, data retrieval, recommendation display) under normal conditions.
    - Peak load: The system should be able to manage up to 500 transactions per minute during peak times (e.g., evening hours).
- **b) Amount of Data to be Processed**:
    - The system must handle user data processing (including responses, session data, and recommendations) in real-time.
    - During peak times, the system should process up to 5,000 user queries without latency exceeding 3 seconds per interaction.

14.3 Response Time:

- 95% of user interactions (questionnaire submissions, recommendation retrieval) should be completed in under 2 seconds.

- 99% of user interactions should be completed in under 3 seconds.

14.4Throughput:

- 95% of data transactions (retrieving user responses and updating records) should be processed within 5 seconds.

15. Logical Database Requirements

The database for the Prakriti-based Chatbot Web Application must be designed to handle a variety of information related to user interactions, user profiles, Ayurvedic recommendations, and system logs. The logical database requirements are as follows:

a) Types of Information Used by Various Functions:

- **User Data**:
    - Name, email address, date of birth, gender, and location.
    - Responses to chatbot questions related to personal habits, preferences, and health data.
    - Results of the Prakriti analysis (Vata, Pitta, Kapha).
    - Personalized Ayurvedic recommendations for food, lifestyle, and yoga practices.
- **Admin Data**:
    - Admin login credentials, role, and permissions for content management.
    - System logs and usage statistics.
    - Feedback from users for improving the system.
- **Prakriti Data**:
    - Predefined Ayurvedic principles (e.g., dosha imbalances, Ayurvedic diet plans).
    - Seasonal and weather-based health tips (if applicable).

b) Frequency of Use:

- **User Data**:
    - High frequency of access during user interaction (daily).
    - Low frequency of updates (user profile modifications, health data updates).
- **Admin Data**:
    - Low frequency of access (primarily for system updates or content management).
- **Prakriti Data**:

- Moderate frequency of access (used for generating recommendations on a per-session basis).

c) Accessing Capabilities:

- **User Data**:
  - Users can access their own data (profile information and recommendations) after logging into the system.
  - Admins have full access to user data (with privacy regulations in place).
- **Admin Data**:
  - Admins will have read/write access to this data for system maintenance, content updates, and error monitoring.
- **Prakriti Data**:
  - Accessed by the system to generate personalized recommendations for users based on their responses and dosha analysis.

d) Data Entities and Their Relationships:

- **Entities**:
  - **User**: Contains personal details, responses to the chatbot, and their Prakriti analysis.
  - **Prakriti**: Stores the dosha classification and corresponding Ayurvedic recommendations (food, lifestyle, yoga).
  - **Admin**: Stores admin login credentials, roles, and permissions.
  - **Feedback**: Contains user feedback and suggestions for system improvement.
- **Relationships**:
  - A User can have one Prakriti analysis (one-to-one relationship).
  - An Admin can manage multiple Users and Prakriti records (one-to-many relationship).
  - A User may provide multiple Feedback entries (one-to-many relationship).

e) Integrity Constraints:

- **Data Integrity**:
  - Ensure that User records are unique based on email addresses.
  - Prakriti data must be consistent with the Ayurvedic principles, ensuring no incorrect dosha analysis is stored.
  - Admin accounts must have strong password policies and proper role-based access control.
- **Referential Integrity**:

- The User entity must reference a valid Prakriti record.
- Feedback must be linked to a User who submitted it.

f) Security:

- **Data Encryption**:
  - Sensitive information like user passwords and health data should be encrypted both at rest and in transit (e.g., using SSL/TLS encryption for user interactions).
- **Access Control**:
  - Admins should have role-based access control (RBAC) to limit access to sensitive information.
  - Users' personal and health data should only be accessible by themselves and administrators, with data sharing restrictions.
- **Audit Logs**:
  - The system must log all access and modification events related to user data and admin activities for security and troubleshooting purposes.

g) Data Retention Requirements:

- **User Data**:
  - Personal information and health data will be retained as long as the user account is active. When a user deletes their account, all associated data should be removed after a period of 30 days.
- **Prakriti Data**:
  - The Prakriti analysis results and recommendations should be retained for at least 1 year to allow users to track changes over time.
- **Admin and System Logs**:
  - Logs should be retained for at least 6 months to comply with system maintenance and troubleshooting needs.

16. Design Constraints

Design constraints refer to the limitations and standards imposed on the software design by external regulations, project limitations, or external systems.

External Standards and Regulatory Requirements:

- **Data Privacy Regulations**:

  - The system must comply with GDPR (General Data Protection Regulation) for users in Europe, ensuring that user data is processed, stored, and deleted according to privacy laws.
  - For Indian users, the system should comply with the Personal Data Protection Bill.

- **Healthcare Regulations**:
  - The system must avoid making medical claims or offering diagnosis-related advice, as this would require specific certifications and licenses.
  - The health and wellness tips must only be considered general advice, and a disclaimer should be presented to users stating that they should consult a medical professional for serious health concerns.

Project Limitations:

- **Technology Stack**:
  The design must utilize the Python Django framework for backend development, React.js for the frontend, and PostgreSQL for the database. The project must also be designed for web-based access, and mobile support should be considered for future iterations.
- **Cloud Hosting**: The system should be deployable on cloud platforms like AWS or Azure, ensuring scalability to accommodate user load growth.
- **Budget and Time Constraints**:
  - The project should be completed within **6 months** from the start date, and the design must be simple enough to ensure that implementation can be done within the allocated budget.
- **User Interface**:
  - The design should ensure a user-friendly interface, with minimal complexity and no more than **5 steps** for a user to complete their Prakriti analysis and receive personalized recommendations.

17. Standards Compliance

The Prakriti-based Chatbot Web Application must comply with the following standards and regulations to ensure smooth operation, security, and compliance with industry norms:

a) Report Format:

- All reports generated by the system (e.g., user activity logs, system errors, and feedback) must follow a standardized format, such as **CSV** or **PDF**, to ensure compatibility with external systems and ease of integration.
- Reports should include clear timestamps, user identifiers, and any relevant details about system actions or user interactions.

b) Data Naming:

- Data entities and attributes must adhere to a standardized naming convention, ensuring clarity and consistency across the system. This includes using descriptive names such as User_ID, Prakriti_Type, and Recommendation_Severity.
- For database table names, use snake_case (e.g., user_profiles, prakti_analysis).
- Field names should be meaningful and avoid abbreviations unless widely recognized in the domain.

c) Accounting Procedures:

- No specific accounting procedures are required as the system does not involve financial transactions. However, user data access should be recorded and processed according to privacy laws (GDPR, Personal Data Protection Bill) to ensure transparency in data handling.
- Any billing, subscription, or donation services should follow standard payment gateway protocols (e.g., integration with Stripe or PayPal).

d) Audit Tracing:

- Audit logs must be implemented to trace all important user and admin actions, such as:
    - User account creation or deletion.
    - Changes to Prakriti analysis or personal health data.
    - Admin access and modifications to user records.
- Logs must capture before and after values, including timestamps, user actions, and the specific changes made to the database (e.g., User_Profile updates).
- All audit traces must be stored securely, with restricted access for admins only, and retained for at least 6 months for compliance with data protection regulations.

18. Software System Attributes

The following attributes are specified for the Prakriti-based Chatbot Web Application to ensure that it meets the desired standards for reliability, availability, security, maintainability, and portability:

a) Reliability:

- Uptime: The system must achieve a minimum of 99.5% uptime to ensure reliable service for users.
- Error Handling: The system should be able to detect, log, and recover from errors without significant disruption to the user experience.
- Fault Tolerance: The application should be designed to handle unexpected failures or downtime gracefully, with appropriate failover mechanisms in place for critical components (e.g., database or server outages).

b) Availability:

- System Availability: The system must be available 24/7, with a planned downtime for maintenance and updates limited to less than 4 hours per month.
- Checkpointing and Recovery: The system must include automated checkpointing and data backup procedures to allow for recovery after an unexpected failure, ensuring minimal data loss (ideally no more than 5 minutes of data).
- Restart Mechanism: In the event of failure, the system should automatically restart, restoring services without manual intervention, and maintaining the session state for users where possible.

c) Security:

- Cryptographic Techniques: The application must use SSL/TLS encryption for all data transmitted over the network, ensuring secure communication between the client and server.
- Access Control: Strong role-based access control (RBAC) mechanisms must be in place to limit access to sensitive data based on user roles (e.g., admin, user, guest).
- Audit Logs: The system should maintain secure audit logs of all critical actions (e.g., user login, data changes) to support traceability and compliance.
- Data Integrity: Integrity checks must be implemented for critical variables, such as user profiles and Prakriti analysis results, to prevent unauthorized data modification.
- Data Privacy: User data should be stored and processed in compliance with GDPR or applicable data protection laws. Personally identifiable information (PII) should be encrypted at rest, and users should have the ability to request data deletion or modification.

d) Maintainability:

- Modularity: The software should be modular, with clearly defined components and interfaces between them, to ease maintenance and upgrades.
- Complexity Limitation: The software should be designed with a low complexity to ensure ease of updates, bug fixing, and testing. Code should be well-documented and adhere to coding standards.
- Automated Testing: Implement automated testing frameworks to support continuous integration (CI) and continuous delivery (CD) pipelines, ensuring faster bug detection and resolution.

e) Portability:

- Operating System Independence: The application should be developed in a platform-agnostic way, supporting major operating systems such as Windows, macOS, and Linux.

- Proven Portable Language: The system should be built using languages that ensure portability, such as Python (for backend) and React (for frontend), which are widely supported across different platforms.
- Hosting Flexibility: The system should be designed to be deployable on cloud platforms such as AWS, Google Cloud, or Microsoft Azure, and should support Docker containers for consistent deployment across different environments.

19. Verification

The following verification approaches and methods will be applied to ensure the quality and reliability of the Prakriti-based Chatbot Web Application:

19.1 Verification Approaches:

- **Unit Testing:** Each component of the system (e.g., backend functions, user input processing) will be tested in isolation to verify its individual functionality. Automated unit tests will be developed using frameworks such **as PyTest (for Python) and Jest** (for React).
- **Integration Testing:** The interaction between components, such as user inputs, backend data processing, and database queries, will be verified through integration tests to ensure the system works as a whole.
- **System Testing:** The complete system will undergo end-to-end testing to validate the software's overall functionality against the specified requirements. This includes performance, security, and usability checks.
- **Acceptance Testing:** The system will undergo User Acceptance Testing (UAT**)** with a select group of end users. Based on their feedback, the system will be refined to meet user expectations and operational needs.
- **Regression Testing:** After each change or update, regression tests will be run to ensure that the new changes do not negatively impact existing functionality.
- **Load and Stress Testing:** The system will be tested under varying loads to ensure it performs optimally under normal and peak conditions. Stress testing will verify that the system can handle extreme or unexpected usage scenarios without crashing.
- **Security Testing:** Security testing, including penetration testing and vulnerability scanning, will be carried out to identify potential security risks (e.g., SQL injection, XSS, CSRF) and validate encryption methods.
- **Compliance Testing:** The system will be verified for compliance with regulatory standards such as GDPR, ensuring proper data protection and user privacy protocols.

19.2 Verification Methods:

- **Manual Review:** The code and documentation will be periodically reviewed by developers and QA engineers to ensure alignment with the requirements.
- **Automated Test Suites:** Continuous integration pipelines will be set up to automatically run tests on the codebase for every commit or change.

- **Code Coverage Analysis:** Tools like SonarQube will be used to ensure sufficient code coverage for testing, ensuring all critical paths are validated.

20. Supporting Information

The following supporting information will be included to assist in understanding and verifying the software requirements:

a) Sample Input/Output Formats:

- **Input Formats:** Examples of user input for the Prakriti chatbot, such as answers to questions about lifestyle, food preferences, and medical history.
- **Output Formats:** Examples of chatbot outputs, including Prakriti prediction (Vata, Pitta, Kapha) and personalized advice (lifestyle, food recommendations).

b) Supporting/Background Information:

- **Background on Prakriti Analysis:** A brief description of the Ayurvedic concept of Prakriti (doshas) and how the chatbot uses user responses to determine the dosha.
- **Target Users:** Information on the target users of the chatbot (e.g., people interested in holistic health and Ayurvedic practices).

c) Problem Description:

- **Problem to be Solved:** The software aims to provide personalized Ayurvedic health advice based on an individual's Prakriti (dosha), helping users improve their lifestyle, food habits, and health choices.

d) Special Packaging Instructions:

- **Code Packaging:** The software code will be packaged in a manner that supports secure distribution, using standard formats such as zip files or Docker containers. This ensures easy deployment and scalability across different environments.
- **Security Requirements for Code Distribution:** Code will be encrypted for secure transmission and storage, with access restricted to authorized personnel only. Necessary access control mechanisms will be implemented to comply with security and export regulations.

Chatbot to known Individual Prakriti (Phenotype)