Technology Justification Report –

AharaSutra

The development of **AharaSutra** relies on a carefully selected technology stack. Each tool, framework, and service has been chosen after evaluating available alternatives in terms of scalability, performance, ecosystem support, security, and suitability for handling large-scale personalized diet plans, Ayurvedic health data, and AI-driven nutrient analysis.

Programming Languages

Java

- **Why chosen:** Java powers the backend and ensures enterprise-grade reliability and security. With Spring Boot, it enables REST APIs for diet chart generation, AI integration, and patient data management. Multi-threading support ensures efficient handling of concurrent users (dietitians and patients).
- **Alternatives considered:**
    - **C# / .NET:** Less commonly used in Indian healthcare startups; licensing may be restrictive.
    - **Node.js:** Lightweight but not ideal for complex business logic and multi-threaded processing.
- **Conclusion:** Java + Spring Boot provides a robust, scalable, and maintainable backend solution for AharaSutra.

Python

- **Why chosen:** Python is used for AI-powered modules like food recommendation, dosha analysis, and nutrient calculations. Libraries such as Pandas, NumPy, and Scikit-learn simplify AI-based personalized diet chart generation.
- **Alternatives considered:**
    - **R:** Strong for statistics but harder to integrate with web-based backend services.
    - **MATLAB:** Proprietary, expensive, and not suitable for cloud deployment.
- **Conclusion:** Python enables rapid AI development, seamless integration with the backend, and open-source flexibility.

Frameworks

React.js

- **Why chosen:** React allows a modular, dynamic, and mobile-friendly frontend. Dietitian dashboards, patient interfaces, quizzes, and interactive diet charts can be built as reusable components.
- **Alternatives considered:**
    - **Angular:** More complex and heavier for small-to-medium projects.
    - **Vue.js:** Lightweight but smaller enterprise adoption.
- **Conclusion:** React ensures scalability, responsiveness, and maintainability of frontend interfaces.

Spring Boot

- **Why chosen:** Spring Boot offers a secure, enterprise-ready backend with REST API support, database integration, and multi-threaded request handling. It facilitates diet chart generation, AI service integration, and secure handling of patient data.
- **Alternatives considered:**
    - **Django:** Python-based but less suited for large-scale multi-threaded backend services.
    - **Flask:** Lightweight but requires manual setup for enterprise-grade features.
- **Conclusion:** Spring Boot is optimal for performance, security, and maintainability in healthcare applications.

Flask

- **Why chosen:** Flask is used for lightweight AI microservices like dosha detection, food recommendation, and nutrient analysis. Minimal setup allows rapid prototyping and integration with Spring Boot backend.
- **Alternatives considered:**
    - **FastAPI:** More performant but adds extra complexity in integration.
- **Conclusion:** Flask simplifies AI module deployment without adding overhead.

Databases

PostgreSQL

- **Why chosen:** PostgreSQL efficiently handles structured patient data, diet charts, and nutritional databases. Supports JSON and indexing for flexible patient record storage, and ensures robust, secure, and scalable storage for long-term use.
- **Alternatives considered:**
    - **MySQL:** Works for relational data but weaker analytics and JSON support.
    - **MongoDB:** Better for unstructured data but inefficient for complex joins and aggregations.
- **Conclusion:** PostgreSQL ensures reliability, scalability, and performance for structured health and nutrition data.

Visualization & Reporting Tools

JasperReports

- **Why chosen:** Automates creation of weekly diet charts in PDF format. Supports tables, charts, and formatted reports ready for download or sharing with patients.
- **Alternatives considered:**
    - **iText:** Requires more manual coding for complex tables.
    - **Crystal Reports:** Proprietary, less flexible for web integration.
- **Conclusion:** JasperReports simplifies report generation and ensures high-quality structured output.

Cloud Infrastructure

AWS

- **Why chosen:** AWS provides scalable hosting, secure storage, and managed database services. Supports auto-scaling for concurrent dietitians and patients, ensuring high availability and global access.
- **Alternatives considered:**
    - **Azure:** Higher cost for startups.
    - **GCP:** Less adoption in Indian healthcare and AYUSH projects.

- **Conclusion:** AWS ensures scalable, secure, and globally accessible cloud infrastructure.

APIs & Tools

REST APIs

- **Why chosen:** REST APIs connect React frontend with Spring Boot backend and Python AI modules. Enables modular architecture for future scalability.
- **Alternatives considered:**
    - **GraphQL:** Flexible for complex queries but adds unnecessary complexity for current project scope.
- **Conclusion:** REST APIs provide a simple, standard, and maintainable integration method.

FastAPI

- **Why chosen:** Handles AI-powered recommendation services asynchronously, ensuring low-latency responses and high performance. Supports automatic documentation for collaborative development.
- **Alternatives considered:**
    - **Flask only:** Synchronous processing may slow down high-volume requests.
- **Conclusion:** FastAPI improves performance for AI modules while simplifying maintenance.