**Software Requirements Specification (SRS) for Scanova**

**1. Introduction**

**1.1 Purpose**

This SRS document outlines the requirements for developing "Scanova," a smart healthcare platform designed to provide cost-effective, efficient, and accessible medical screening and personalized healthcare solutions. The platform leverages AI-driven technologies to address global healthcare challenges such as delayed disease detection, fragmented data, and limited access to advanced diagnostics. This document serves as a blueprint for developers to build a fully functional website with all specified services, pages, and integrations.

**1.2 Scope**

Scanova aims to revolutionize healthcare by offering:

* Early disease detection using AI-driven voice and image analysis.
* Accessibility for rural and underserved areas through cloud-based remote diagnostics and consultations.
* Cost-effective healthcare by reducing reliance on expensive tests.
* Unified patient data management with secure cloud storage.
* Personalized healthcare recommendations via an AI chatbot.

The website will include a homepage, dashboard, doctor connection module, smart screening services, medical records management, and an AI-powered personal assistant. Future enhancements will expand diagnostic capabilities and integrate telemedicine features.

**1.3 Intended Audience**

* **End Users**: Patients, especially in rural and underserved areas, seeking accessible healthcare.
* **Doctors**: Healthcare professionals providing remote consultations.
* **Developers**: Teams building and maintaining the platform.
* **Stakeholders**: Project team from Sinhgad College of Engineering (Azfar Mohammad Javed Shaikh, Yash Laxmikant Kasat, Isha Katariya, Prasad Bhagvawant, Sanket Palve).

**1.4 Definitions**

* **AI**: Artificial Intelligence
* **Scanova**: The smart healthcare platform described herein.
* **Smart Screening**: AI-driven diagnostic tools for early disease detection.
* **LVM**: Latent Variable Model (assumed AI model for diagnostics).

**2. Overall Description**

**2.1 User Needs**

* Patients need affordable, accessible, and timely healthcare solutions.
* Doctors require a platform to connect with patients remotely and access unified medical records.
* Rural communities need diagnostics without geographical or financial barriers.

**2.2 Assumptions and Dependencies**

* Users have access to a stable internet connection for real-time features.
* Compatible devices (e.g., smartphones, wearables) are available for voice and image inputs.
* IBM Watson AI, Auto AI, and other third-party services are accessible via APIs.

**2.3 Product Perspective**

Scanova is a web-based platform integrating AI-driven diagnostics, cloud storage, and telemedicine features. It connects patients and doctors, automates health screenings, and provides personalized insights, addressing inefficiencies in traditional healthcare systems.

**3. System Features**

**3.1 Homepage**

* **Description**: A user-friendly landing page introducing Scanova’s services and features.
* **Functionalities**:
  + Navigation menu linking to Dashboard, Doctors, Services, Medical Records, and Personal Assistant.
  + Brief overview of Scanova’s mission and benefits.
  + Call-to-action buttons (e.g., "Get Started," "Learn More").
* **Input**: User clicks.
* **Output**: Redirects to respective pages.

**3.2 Dashboard**

* **Description**: A personalized interface for users to monitor health insights and access tools.
* **Functionalities**:
  + Real-time health updates (e.g., recent screening results).
  + Quick access to smart screening tools and consultation booking.
  + Visual charts summarizing health data.
* **Input**: User login, health data inputs (voice, images).
* **Output**: Personalized health summary and navigation options.

**3.3 Doctors Module**

* **Description**: Connects users with doctors for remote consultations.
* **Functionalities**:
  + Doctor profiles (specialization, availability).
  + Real-time video consultation scheduling and integration (e.g., WebRTC).
  + Secure sharing of medical records with doctors.
* **Input**: User selects doctor, schedules consultation.
* **Output**: Confirmation of booking, video call interface.

**3.4 Smart Screening Services**

* **Description**: AI-driven tools for early disease detection.
* **Functionalities**:
  + **Voice-Based Health Check-up**: Analyzes voice samples for diseases (e.g., Parkinson’s) using IBM Watson Speech-to-Text and AI models.
  + **Image-Based Screening**: Detects diseases (e.g., skin cancer, lung cancer, tuberculosis) via image uploads processed by AI (e.g., LVM Model).
  + Results displayed with confidence scores and recommendations.
* **Input**: Voice recordings, medical images (e.g., X-rays, skin photos).
* **Output**: Diagnostic report with health insights.

**3.5 Medical Records Management**

* **Description**: Secure storage and management of health records.
* **Functionalities**:
  + Upload and organize medical documents (prescriptions, reports, images).
  + Cloud-based access for patients and authorized doctors.
  + Search and filter options for records.
* **Input**: File uploads (PDFs, images, text).
* **Output**: Organized, accessible records.

**3.6 Personal Assistant**

* **Description**: An AI chatbot for personalized healthcare guidance.
* **Functionalities**:
  + Provides tailored health recommendations based on screening results.
  + Answers user queries about health conditions and platform usage.
  + Integrates with IBM Watson AI for natural language processing.
* **Input**: Text or voice queries.
* **Output**: Text or voice responses with guidance.

**4. External Interface Requirements**

**4.1 User Interfaces**

* Responsive design using TypeScript + Vite Framework for frontend.
* Intuitive navigation with minimal clicks to access features.
* Visual feedback (e.g., loading animations, success messages).

**4.2 Hardware Interfaces**

* Compatible with smartphones, laptops, and wearables for voice/image inputs.
* Requires microphone and camera for voice and image-based screenings.

**4.3 Software Interfaces**

* **Frontend**: TypeScript, Vite Framework.
* **Backend**: Python (e.g., Flask/Django for API development).
* **Database**: IBM DB2 for storing user data and medical records.
* **AI Services**: IBM Watson AI (Speech-to-Text, NLP), Auto AI, LVM Model for diagnostics.
* **Cloud Storage**: Secure cloud service (e.g., IBM Cloud, AWS S3) for records.

**4.4 Communication Interfaces**

* HTTPS for secure data transmission.
* WebRTC for real-time video consultations.
* RESTful APIs for integrating AI services and database.

**5. Non-Functional Requirements**

**5.1 Performance**

* Process voice/image inputs and return results within 10 seconds.
* Support up to 10,000 concurrent users without latency.

**5.2 Security**

* Encrypt all medical data (e.g., AES-256).
* Implement user authentication (e.g., OAuth 2.0).
* Regular security audits to prevent breaches.

**5.3 Reliability**

* 99.9% uptime for cloud-based services.
* Backup and recovery mechanisms for data loss prevention.

**5.4 Usability**

* Accessible to non-technical users (e.g., elderly, rural populations).
* Multilingual support for broader reach.

**5.5 Scalability**

* Modular design to add new screening tools and features.
* Cloud infrastructure to handle increased user load.

**6. System Architecture**

**6.1 Overview**

* **Frontend**: TypeScript + Vite for a fast, responsive UI.
* **Backend**: Python-based server handling API requests and business logic.
* **Database**: IBM DB2 for structured data (user profiles, records).
* **AI Layer**: IBM Watson AI, Auto AI, and LVM Model for diagnostics.
* **Cloud**: IBM Cloud/AWS for storage and scalability.

**6.2 Data Flow**

1. User inputs (voice, images) → Frontend → Backend API.
2. Backend processes inputs → AI services → Diagnostic results.
3. Results stored in DB → Displayed on Dashboard/Personal Assistant.

**7. Integration Requirements**

**7.1 Third-Party APIs**

* **IBM Watson Speech-to-Text**: For voice analysis.
* **IBM Auto AI**: For predictive modeling.
* **WebRTC**: For video consultations.
* **Cloud Storage API**: For secure document management.

**7.2 Hardware Integration**

* Microphone and camera support for input collection.
* Optional wearable device integration (e.g., heart rate monitors).

**8. Future Scope**

**8.1 AI-Powered Diagnostic Tools**

* Expand to detect neurological (e.g., Alzheimer’s) and cardiovascular diseases.
* Integrate wearable data for continuous monitoring.

**8.2 Telemedicine Enhancements**

* Add virtual prescriptions and follow-up scheduling.
* Integrate with pharmacies for medicine delivery.

**8.3 Global Expansion**

* Support additional languages and regional healthcare standards.
* Partner with local healthcare providers in underserved regions.

**9. Deliverables**

* Fully functional website with all pages (Homepage, Dashboard, Doctors, Services, Medical Records, Personal Assistant).
* Source code hosted on Replit with documentation.
* API endpoints for all functionalities.
* User manual for patients and doctors.

Prompt for replit ai :

Build a complete website for "Scanova," a smart healthcare platform, based on the provided SRS. Use TypeScript + Vite for the frontend, Python (Flask/Django) for the backend, and IBM DB2 for the database. Integrate IBM Watson AI (Speech-to-Text, NLP), Auto AI, and LVM Model for diagnostics, and use WebRTC for video consultations. Implement the following:

1. Homepage: Responsive landing page with navigation and overview.

2. Dashboard: Personalized health insights with charts and tool access.

3. Doctors Module: Doctor profiles and video consultation scheduling.

4. Smart Screening Services: Voice-based (Parkinson’s) and image-based (cancer, TB) diagnostics.

5. Medical Records: Secure cloud storage and retrieval system.

6. Personal Assistant: AI chatbot for recommendations and queries.

Ensure security (encryption, authentication), scalability, and a modern UI. Include API documentation and a user manual. Simulate AI integrations if real APIs are unavailable. Deploy the project on Replit with all functionalities working as described.