Software Requirements Specification (SRS)

# Project Title: MediBridge

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to detail the software requirements for MediBridge, a medical dashboard designed to streamline communication between hospital staff and patients, and to assist in managing medical history, alerts, and AI-driven interaction.

## 1.2 Scope

MediBridge aims to provide an integrated platform for hospital administrators and patients. It offers functionalities such as:  
- Patient medical history tracking  
- Automated alert system for real-time updates  
- AI chatbot for instant patient assistance  
This system helps enhance patient experience and assists hospitals in managing patient data efficiently.

## 1.3 Definitions, Acronyms, and Abbreviations

- AI – Artificial Intelligence  
- UI – User Interface  
- DB – Database  
- MediBridge – Name of the medical dashboard application

## 1.4 References

- Project Repository: https://github.com/Karanpratap7/PBL/tree/main  
- Node.js Official Docs: https://nodejs.org  
- Express.js Docs: https://expressjs.com  
- MongoDB Docs: https://www.mongodb.com/docs/

## 1.5 Overview

This document outlines both functional and non-functional requirements, system architecture, interfaces, and constraints of the MediBridge system.

# 2. Overall Description

## 2.1 Product Perspective

MediBridge is a web-based application built using Node.js, Express.js, and MongoDB. The frontend is built using HTML, CSS, and JavaScript. It is designed to be modular and scalable for integration with hospital systems.

## 2.2 Product Functions

- Maintain and access patient medical history  
- Notify users (admin/patient) through alerts (e.g., appointment reminders, emergency updates)  
- Interact with users using an AI chatbot to resolve common queries

## 2.3 User Classes and Characteristics

- Hospital Admins: Manage patient data, configure alerts, view dashboard metrics  
- Patients: View personal health data, receive alerts, interact with chatbot

## 2.4 Operating Environment

- Web Browsers: Chrome, Firefox, Edge  
- Server: Node.js runtime on backend  
- Database: MongoDB (NoSQL)  
- Hosting: Can be deployed on platforms like Render, Heroku, or any VPS

## 2.5 Constraints

- Internet access required for real-time chatbot and dashboard features  
- Compatibility limitations with outdated browsers  
- Scalability dependent on backend hosting and DB limits

## 2.6 Assumptions and Dependencies

- Admins have basic technical proficiency  
- AI chatbot API is available and integrated  
- MongoDB server is configured and running

# 3. Specific Requirements

## 3.1 Functional Requirements

FR1: User Authentication  
- Patients and Admins must log in using valid credentials  
- Different dashboards are shown depending on the user role  
  
FR2: Patient Dashboard  
- View medical history  
- Access AI chatbot  
- Receive health alerts  
  
FR3: Admin Dashboard  
- Add/update patient data  
- Configure alerts  
- Monitor chatbot analytics (if available)  
  
FR4: Alert System  
- Scheduled and conditional alerts  
- Email/SMS/Push notifications (optional)  
  
FR5: AI Chatbot  
- Understand and answer health-related queries  
- Direct users to the appropriate dashboard section

## 3.2 Non-Functional Requirements

- Performance: Responses must be provided in under 2 seconds  
- Scalability: Able to support multiple simultaneous users  
- Security: Data encryption, secure login, and access control  
- Usability: Clean UI/UX for both admin and patient roles  
- Availability: 99% uptime if deployed on a production server

## 3.3 External Interface Requirements

### 3.3.1 User Interfaces

- Web-based UI with responsive design  
- Dashboard for patients and admins  
- Chat interface for AI bot

### 3.3.2 Hardware Interfaces

- None required beyond standard computing devices

### 3.3.3 Software Interfaces

- Node.js + Express.js backend APIs  
- MongoDB for database operations  
- AI API for chatbot (could be OpenAI, Dialogflow, etc.)

# 4. Appendix

- Future Enhancements: Appointment booking, health tips section, analytics reports  
- Development Tools: VSCode, Git, Postman, MongoDB Compass  
- Testing Tools: Mocha/Chai or Jest for backend tests