SOFTWARE REQUIREMENT SPECIFICATION (SRS)

1. Introduction

1.1 Purpose

The purpose of this document is to describe the requirements of the React + Node.js Mini Project.
This project extends an existing React application by integrating a Node.js backend to handle all data operations through APIs.

1.2 Scope

The system is a full-stack web application where:

- React is used for the frontend
- Node.js and Express are used for the backend
- MongoDB Atlas is used for persistent data storage

2. Problem Statement

Frontend-only applications cannot store data permanently or handle real-world scenarios efficiently.
This project solves that problem by integrating a backend server and database, enabling secure data storage, retrieval, updating, and deletion using REST APIs.

3. Objectives

- To convert a frontend-only React application into a full-stack application
- To implement RESTful APIs using Node.js and Express
- To store application data permanently using MongoDB Atlas
- To demonstrate CRUD (Create, Read, Update, Delete) operations
- To integrate frontend and backend using asynchronous API calls

4. Functional Requirements

4.1 User Requirements

- Users should be able to submit contact details through a form

- Users should be able to submit admission enquiries
- Submitted data should be stored in the database
- Users should receive success or error messages after submission

4.2 System Requirements

Contact Module

- Create contact using POST API
- Fetch all contacts using GET API
- Update contact using PUT API
- Delete contact using DELETE API

Admission Module

- Submit admission enquiry using POST API
- Fetch admission enquiries using GET API

5. Non-Functional Requirements

- The system should be responsive and user-friendly
- API responses should be in JSON format
- The backend should handle invalid input gracefully
- The system should ensure data persistence
- The application should be easy to maintain and extend

6. Technology Stack

Frontend

- React.js
- HTML
- CSS
- JavaScript

Backend

- Node.js

- Express.js

Database

- MongoDB Atlas

Tools

- GitHub
- Postman
- VS Code

7. System Architecture

- React frontend communicates with backend using REST APIs
- Node.js backend processes requests and performs validations
- MongoDB Atlas stores contact and admission data
- Data flow:
  Frontend → Backend API → Database → Backend → Frontend

8. API Details

Contact APIs

- POST /api/contacts
- GET /api/contacts
- PUT /api/contacts/:id
- DELETE /api/contacts/:id

Admission APIs

- POST /api/admissions
- GET /api/admissions

9. Testing

- APIs are tested using Postman
- CRUD operations are verified
- Proper success and error responses are validated

10. Conclusion

This project successfully demonstrates a real-world full-stack application using React, Node.js, Express, and MongoDB.
It highlights API-based communication, persistent data storage, and clean separation between frontend and backend.