**SYSTEM DESIGN FOR HOSPITAL MANAGEMENT**

1. **PURPOSE OF THE APPLICATION**

To create, save , delete and display patients and doctors information.

Multiple patients can be assigned to doctor based on the doctor's profile, like cardiac specialist, Physio etc.

**2) REQUIREMENTS OF THE SYSTEM**

****Functional Requirements****

**- User should be able to create or delete patient details**

**- User should be able to create or delete doctor details**

****-** Displaying patient and doctor details**

****Non Functional Requirements****

**-**Our service needs to be highly available

- Acceptable latency should be 200 ms

- System should be highly reliable , any data stored should never lost

1. **DESIGN CONSIDERATION**

- Data should be 100% reliable, if a patient information is stored it should never be lost

- System should be highly available(low latency)

1. **HIGH LEVEL DESIGN**

We need to handle two scenarios data writing and data retrieving

Data writing service

Client

Database

Services

Data reading service

Client

1. **DATABASE SCHEMA**

|  |
| --- |
| **Patient Table** |
| Patient ID PK |
| Doctor ID |
| Patient Name |
| Patient disease |
| Patient history |
| Creation Date |

|  |
| --- |
| **Doctor Table** |
| Doctor ID PK |
| Doctor Name |
| Doctor specialist |
| Doctor availability |
| Creation date |

A straightforward way to store this schema is to use a RDBMS like MYSQL since it requires join but it has its own challenges (scaling)

Doctor ID and Patient ID is form by (auto increment serial number + creation date in EPOCH) that way we can easily sort the data.

**6) COMPONENT DESIGN**

Basically writing data to a DB is slower than reading the data, also sometimes patients details can contain scan reports as images.

Photo uploads (or writes) can be slow as they have to go to the disk, whereas reads will be faster, so when the application goes to real time there might bottleneck when uploading and reading happens in same time(reads cannot be served if the system gets busy with all the ‘write’ requests).To handle this we can split read and write services into separate service and We will have dedicated servers for reads and different servers for writes to ensure that write don’t hog the system it also allows us to scale and optimize each services independently

Data reading service

Data writing service

Client

Client

Read Service

Database

Write Service

Database

**7) RELIABILITY AND REDUNDANCY**

In hospital management system patient details are vital information so losing it isn’t an option.Therefore, we will store multiple copies of the data so that if one server storage dies we can get it from another storage server

Same principle can be applied to the service as well, if we want to have high availability of the system , we need to have multiple replicas of the services running int he system so that even if a few services die down, the system remains available and running

Creating redundancy in a system can remove single points of failure and provide a backup or spare functionality if needed in a crisis. For example, if there are two instances of the same service running in production and one fails or degrades, the system can failover to the healthy copy. Failover can happen automatically or require manual intervention.

Replicated Database

Replicated Database

Database

Replicated Database

Replicated Database

Database

Read Service

Client

Read Service

Data reading service

Write Service

Data writing service

Client

Write Service