**DESIGN PATTERN USED IN OUR APPLICATION**

**MVC DESIGN PATTERN:**

To create Entities,Models and to use it accordingly MVC pattern is the suitable for a spring boot application,also MVS pattern is very light weight implementation and it makes testing of controllers very comprehensive.

**DOMAIN DRIVEN DESIGN**

I have split the application as doctor and patient (class,object, and model) as per domain driven design. So its easy to maintain, enhance each independently.Also have created Rest API based on the domain(doctor or patient) so the URL will be <http://localhost:8080/doctor/detail> for doctor related API and [http://localhost:8080/patient/detail](http://localhost:8080/doctor/detail) for patient related API

**SINGLETON DESIGN PATTERN**

I have used singleton design pattern for exception and error handling which is responsible to create an object while making sure that only single object gets created. Its easy to handle errors and to eliminate code redundancy.

**SOLID PRINCIPLE**

**Open-close principle**

In DoctorRepo am extending CrudRepository interface to extend its function by adding findBySpecialist without changing its base code.

## **Interface segregation principle**

I have split the application as doctor and patient for all(class,object ,view and model). so doctors interface wont affect or depend patient interface, since doctors wont be using patients UI and wise verse