Robot Part Inventory

1)Give A Simple Sketch Of The Solution:-

- The given problem statement is a request/response paradigm design which is taken care under by SOA based solution architecture.
- RESTful Web services are used for interfacing between client and server and Http Request Handler is responsible for handling the requests proposed by the users.

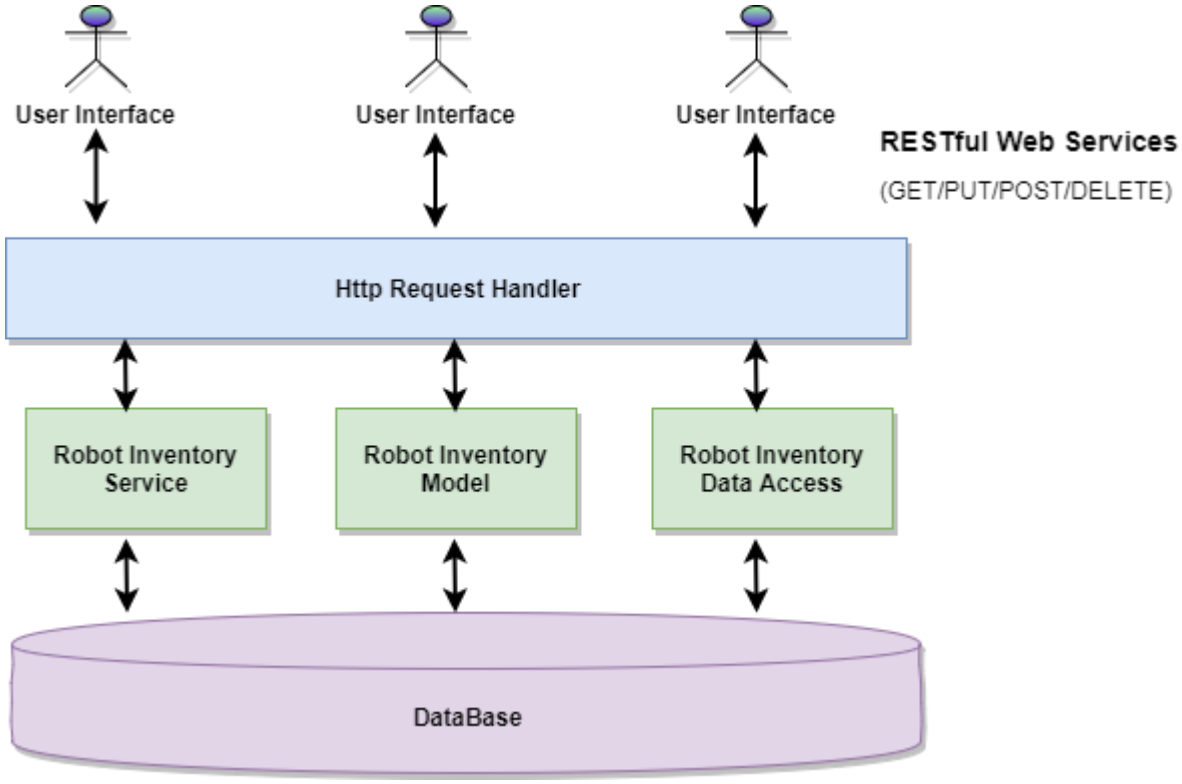

Fig 1.0

2)Outline DataBase Model

- According to the nomenclature of Relational database management system and following the Normalization forms, there are three relations to be made to achieve the give problem statement:-
- Robot Inventory Details
- Compatible Parts Details
- All Parts Details
- In solution to this, Hibernate is used for object relational mapping and database interaction.
- Let us see the Schema relationship of the relations.

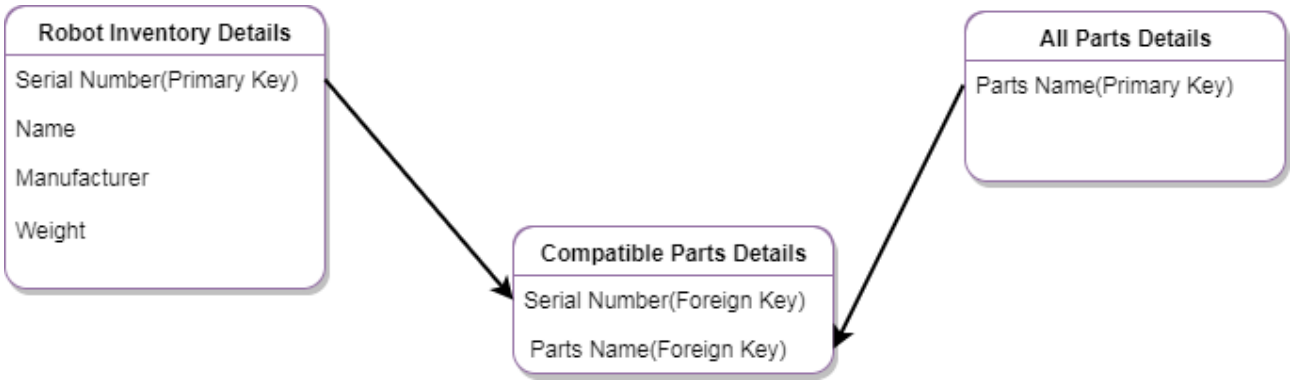

Fig 2.0

3)Implement the Web Service and Client

- The RESTful Web services technique is used for achieving the solution having four requests like GET, POST, PUT and DELETE for performing the Read, Create, Update and Delete operations.
- The client is implemented by basic HTML page with AJAX and Jquery features for request and response handling to Web services.
- Let us have a look about the screens created for client looks like for the given solution:-

Fig3.0 Main Page

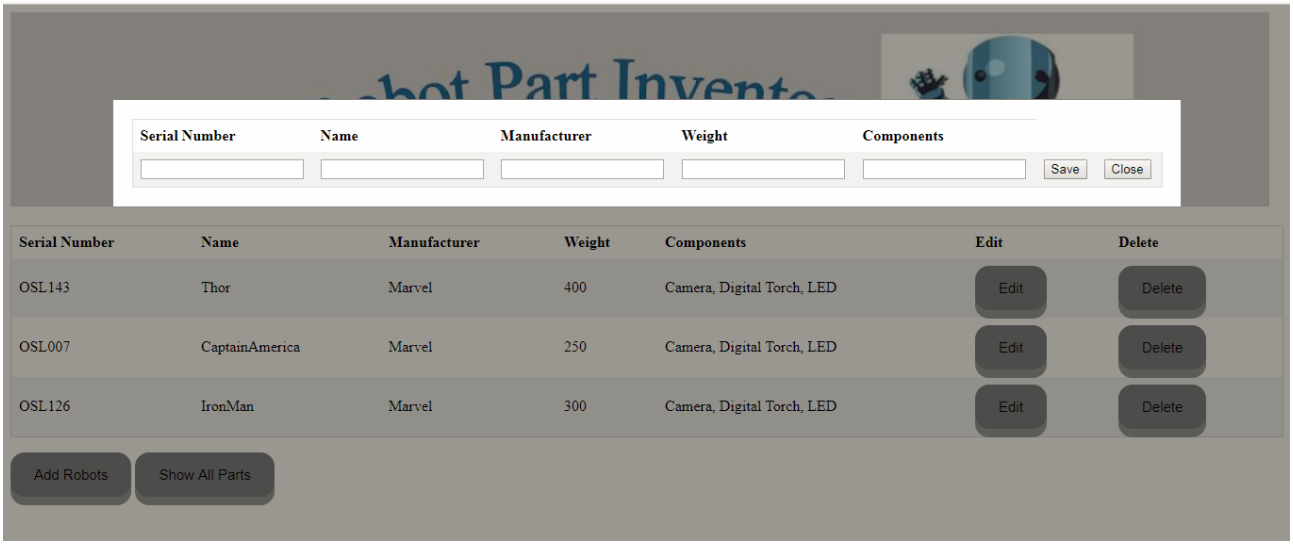

Fig 3.1 Add Robots Page(Click on Add Robots Button)

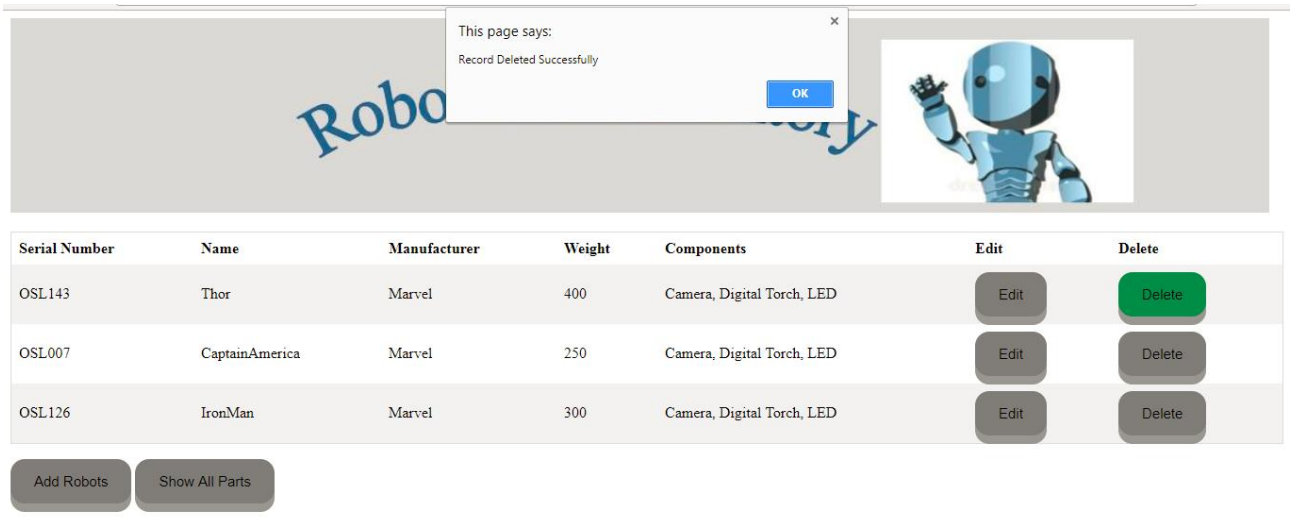

Fig 3.2 Delete Robot Highlighted(Click on Delete Robot Button)

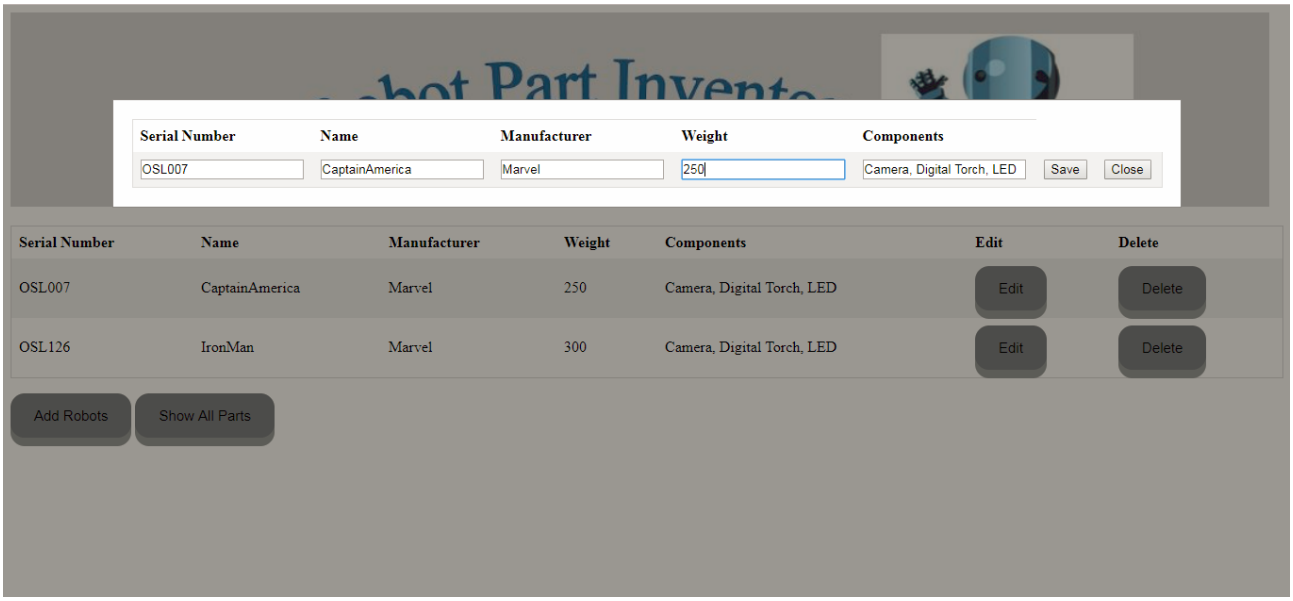

Fig 3.3 Edit Robot Page(Click on Edit Robot Button)

Fig 3.4 All Components List(Click on Show All parts from main Page)

4)Scalability, High Availability and Deployment:

- To achieve the scalability and high availability for the given solution, Following steps needs to be done:-

- The Application would be deployed using one of the containerization technique, We can use **Docker to deploy the application** as a docker image in the docker only.
- All the dependencies for making the application up and running like Tomcat web server, Java, Python etc. would be removed after deploying the application as a docker image in docker.
- To achieve the scalability, **Container orchestration would be used like kubernetes**, that would help to enhance the instance of docker images and creating new instances of the application when one instance is not able to handle all the requests. Kubenetes, in this scenario control the scalability for the system for handling multiple requests at the same time by increasing the number of instances or managing the existing ones.
- To achieve the high availability of the data, **NoSQL database like MongoDB or couchbase** would be used for complex data storing and fast processing of the stored data.

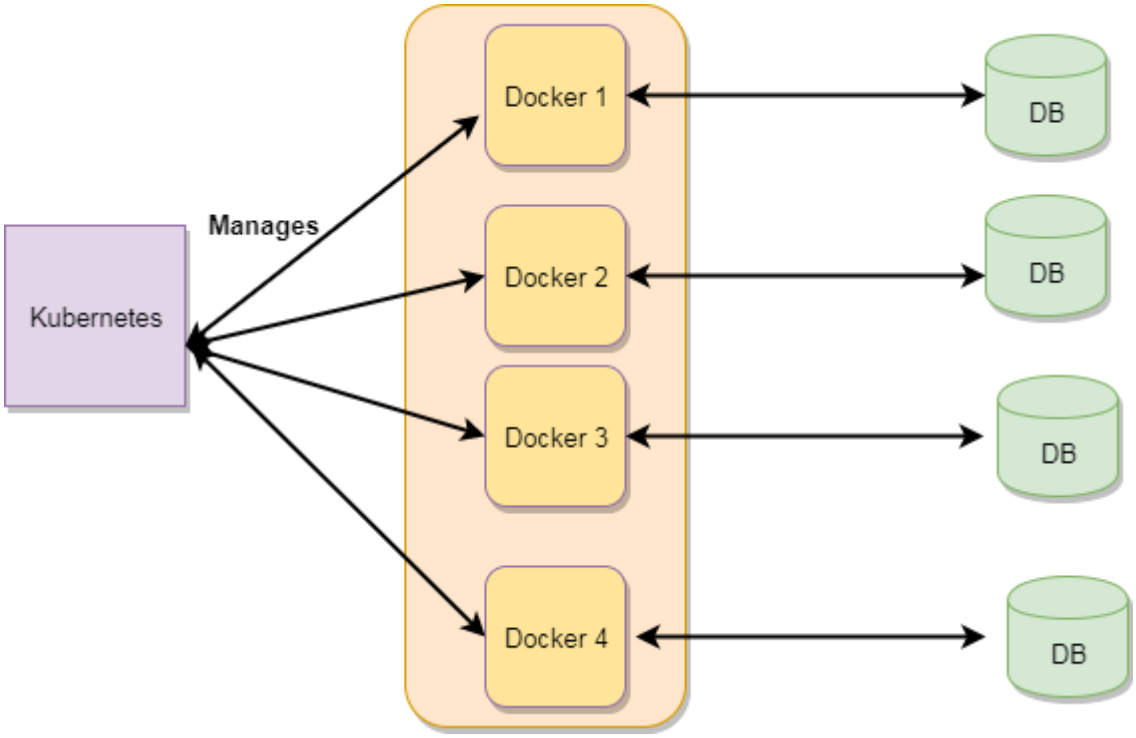

Fig 4.0

5)Securing Web Services:

- The RESTful web services in the given solution is secure by using web.xml deployment descriptor.
- Please find the below screenshot of the deployment descriptor security management implementation:-

```
<security-constraint>
     <web-resource-collection>
          <web-resource-name>Orders</web-resource-name>
          <url-pattern>/orders</url-pattern>
          <http-method>GET</http-method>
          <http-method>POST</http-method>
          <http-method>PUT</http-method>
          <http-method>DELETE</http-method>
     </web-resource-collection>
     <auth-constraint>
          <role-name>admin</role-name>
     </auth-constraint>
</security-constraint>
    <login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>default</realm-name>
    </login-config>
<security-role>
    <role-name>admin</role-name>
</security-role>
```