**Design and Implementation:**

Please find attached FinalProject folder for drpatients web services for detail source code. Frame is used for this project is JAX-RS API using Jersey and predictions3 code base from book is used to develop drpatients web services. There are total four resource classes were used to develop this web services. 1. Patient 2. DrpatientsList 3. RestfulDrpatient 4. DrpatientsRS and two input files patients.db and drs.db are used as input data files. Web service is deployed on Tomcat server using apache Ant automation build tool and curl utility is used for performing some CURD operations like POST, PUT and DELETE.

I had great experience in creating this project. I get good understanding of JAX-RS API using Jersey framework. I had to refresh my knowledge in List to develop CURD operations and handle doctor and patient information. In this project, I have also created CURD operations like GET, POST, PUT and DELETE to support requirements. I have also created some error handling for bad requests. In user manual, I have provided screen shots for each operation which is part of requirements.

In this section, four main resources are discussed with their main operations.

**First, Patient class** has patient information like his/her id, name and insurance card number, and doctor information like his/her id, name and number of corresponding patients. This class has getter and setter methods which will be used for getting and setting data during entire project. It also override toString() method to display output as per requirement.

@Override

public String toString() {

return drwho+String.format("%2d: ", id-1) + whoPat + " ==> " + whatId +"\n";

}

The Patient class implements the interface Comparable.

public class Patient implements Comparable<Patient>

The implementation of the compareTo method means that Patient objects are sorted correctly for CRUD operations.

public int compareTo(Patient other) {

return this.id - other.id;

}

**Second, The DrpatientsList Class** has thread safe CopyOnWriteArrayList which implements Patient Objects with id for each doctor patient list.

public DrpatientsList() {

drpats = new CopyOnWriteArrayList<Patient>();

drpatsId = new AtomicInteger();

}

The DrpatientsList Class has as an @XmlRootElement annotation so that

Jersey automatically serializes a list of doctor and patients.

@XmlElement

@XmlElementWrapper(name = "drpatients")

public List<Patient> getDrpatients() {

return this.drpats;

}

The DrpatientsList class also overrides the toString method to support a plain text display of Patient information.

@Override

public String toString() {

String s = "";

for (Patient p : drpats) s += p.toString();

return s;

}

The DrpatientsList class has find method which finds doctor information based on his/her corresponding id. It iterates through a Patient list and finds a corresponding doctor id. This method is really useful in CURD operations like PUT, and DELETE.

public List<Patient> find(int id) {

List<Patient> dpat=new ArrayList<Patient>();

for (Patient p : drpats) {

if (p.getIddr() == id) {

dpat.add(p);

//break;

}

}

return dpat;

}

The DrpatientsList class has add method which adds patient and doctor information into Patient list object using setter methods from Patient class.

public int add(int drIdval,String numPat,String drwho,String whoPat, String whatId) {

int id = drpatsId.incrementAndGet();

Patient dp = new Patient();

dp.setDrWho(drwho);

dp.setWhoPat(whoPat);

dp.setWhatId(whatId);

dp.setId(id);

dp.setIddr(drIdval);

dp.setNumpat(numPat);

drpats.add(dp);

return id;

}

**Third, The DrpatientsRS class** mainly has four CURD operations such as GET, POST, PUT and DELETE which has annotations like @GET, @POST, @PUT and @DELETE respectively for all doctor patients and specific one. It mainly supports requests in plain and xml format as per requirement. It has populate() method which reads data from two input files patients.db and drs.db and adds parsed doctor and patient information into addDrPatient method which then adds data into instance of DrpatientsList class.

// Add a new drpatients to the list.

private int addDrPatient(int drIdval,String numPat, String drwho,String whoPat, String whatId) {

int id = DrpatList.add(drIdval,numPat,drwho,whoPat, whatId);

return id;

}

CURD operations like PUT, DELETE and GET for specific id uses find method from DrpatientsList class where it finds doctor and patient information based on supplied id and then it performs delete, update or get information for that specific list of doctor and patient information.

List<Patient> drpatlist = DrpatList.find(id);

//DELETE

for (Patient dp: drpatlist){

DrpatList.getDrpatients().remove(dp);

msg = "Doctor Patient Informaiton " + id + " is deleted.\n";

}

//GET Specific doctor and patient information

for (int i=0; i<drpatlist.size(); i++){

drPat=drPat+drpatlist.get(i).toString();

}

//Update Specific doctor information

Patient dP=new Patient();

if (drName != null) dP.setDrWho(drName);

if (patCount != null) dP.setNumpat(patCount);

**Forth, The RestfulDrpatient class** extends the JAX-RS Application class and after its deployment on Tomcat server it invokes a getClasses method to instantiate RESTful resources from WAR file which is DrpatientsRS for drpatients web services.

@Override

public Set<Class<?>> getClasses() {

Set<Class<?>> set = new HashSet<Class<?>>();

set.add(DrpatientsRS.class);

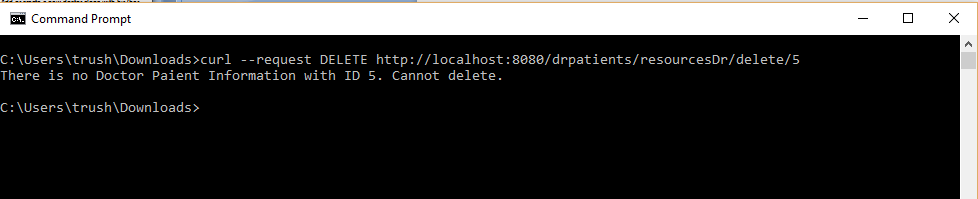
return set;

}

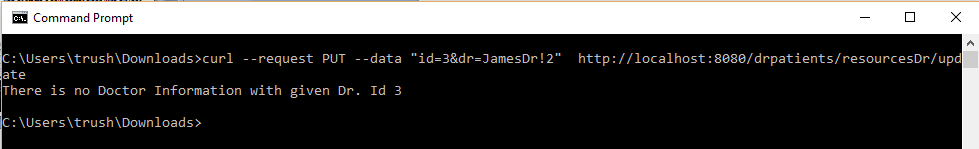
**Handling bad request:**

In this section, we will go over some bad request scenarios which are handle in drpatients webservices.

**Scenario 1:** Delete a doctor who does not exist.



**Scenario 2:** Update a doctor who does not exist.



**Scenario 3:** Update a doctor who does not have same patient count.

