FIT5046 Assignment1

Student name: Qiwei Wang

Student ID: 28840836

Tutor: Himanshu Pahuja

Task 1

a) SQL

User table:

```
create table "WQW".USERINFO
(
        USERID INTEGER not null primary key,
        NAME VARCHAR(30),
        SURENAME VARCHAR(30),
        EMAIL VARCHAR(30),
        DOB DATE,
        HEIGHT INTEGER,
        WEIGHT INTEGER,
        GENDER VARCHAR(10),
        ADDRESS VARCHAR(30),
        POSTCODE VARCHAR(30),
        LEVEL_OF_ACTIVITY INTEGER,
        STEPS_PER_MILE INTEGER,
        CHECK (GENDER = 'male' OR GENDER = 'female'),
        CHECK (LEVEL_OF_ACTIVITY >= 1 AND LEVEL_OF_ACTIVITY <= 5)
)
```

Records in the user table:

#	USERID	NAME	SURENAME	EMAIL	DOB	HEIGHT
1	1	AAA	AAA	123@ABC.COM	1994-04-26	175
2	2	BBB	BBB	234@ABC.COM	1988-03-26	182
3	3	CCC	CCC	345@ABC.COM	1997-11-06	165
4	4	DDD	DDD	132@ABC.COM	2000-04-11	163
5	5	EEE	EEE	432@ABC.COM	1980-07-07	177

Food table:

```
create table "WQW".FOOD
(
        FOODID INTEGER not null primary key,
        FOOD_NAME VARCHAR(30),
        CATEGORY VARCHAR(30),
        CALORIE_AMOUNT INTEGER,
```

SERVING_UNIT VARCHAR(30),

SERVING_AMOUNT DOUBLE,

FAT INTEGER

)

Records in the food table:

#	FOODID	FOOD_NAME	CATEGORY	CALORIE_AMOUNT	SERVING_UNIT	SERVING_AM
1	1	apple butter	jam	34	Tbsp	
2	2	apples, dried	fruit	52	cup	
3	3	apple, 2 3/4″diam	fruit	81	each	
4	4	avocado, black or green skin	fruit	121	cup	
5	5	baby corn	vegetable	20	cup	
6	6	turkey bacon	meat	32	slice	
7	7	pork bacon	meat	36	slice	
8	8	bacon fat	oil	89	Tbsp	
9	9	white bagel, 3″diam	dessert	157	each	
10	10	whole wheat bagel, 3″diam	dessert	168	each	
11	11	vegetarian baked beans	beans	127	cup	
12	12	barbecue sauce	sauce	12	Tbsp	
13	13	canned beef	meat	166	cup	
14	14	fresh blackberries	fruit	37	cup	
15	15	beets	vegetable	37	cup	
16	16	mung beans	beans	139	cup	
17	17	fresh blueberries	fruit	41	cup	
18	18	broccoli	vegetable	26	cup	
19	19	carrots	vegetable	35	cup	
20	20	catsup	sauce	16	Tbsp	

Consumption table:

create table "WQW".CONSUMPTION

(

CONID INTEGER not null primary key,

USERID INTEGER not null,

DATE DATE not null,

FOODID INTEGER not null,

QUANTITY INTEGER not null

)

ALTER TABLE CONSUMPTION

ADD CONSTRAINT FK_CONSUMPTION_USERID

FOREIGN KEY (USERID) REFERENCES USERINFO (USERID)

ON DELETE CASCADE;

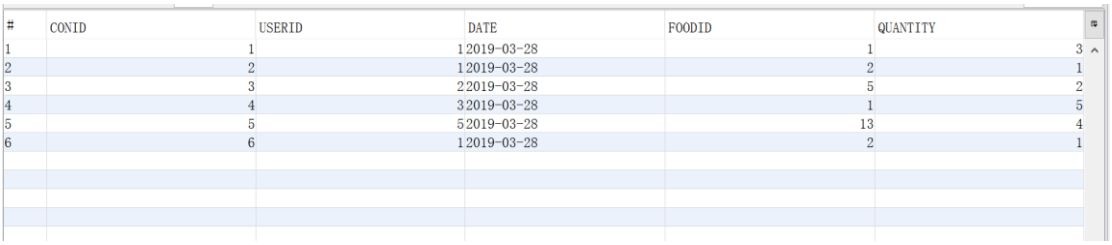
ALTER TABLE CONSUMPTION

ADD CONSTRAINT FK_CONSUMPTION_FOODID

FOREIGN KEY (FOODID) REFERENCES FOOD (FOODID)

ON DELETE CASCADE;

Records in the consumption table:



#	CONID	USERID	DATE	FOODID	QUANTITY
1	1	1	2019-03-28	1	3
2	2	1	2019-03-28	2	1
3	3	2	2019-03-28	5	2
4	4	3	2019-03-28	1	5
5	5	5	2019-03-28	13	4
6	6	1	2019-03-28	2	1

Credential table:

```
create table "WQW".CREDENTIAL
(
    CREID INTEGER not null primary key,
    NAME VARCHAR(30),
    USERID INTEGER not null,
    PASSWORD_HASH VARCHAR(30) not null,
    SIGN_UP_DATE DATE not null
)
ALTER TABLE CREDENTIAL
ADD CONSTRAINT FK_CREDENTIAL_USERID
FOREIGN KEY (USERID) REFERENCES USERINFO (USERID)
ON DELETE CASCADE;
```

Records in the credential table:

#	CREID	NAME	USERID	PASSWORD_HASH	SIGN_UP_DATE
1	1	test1	1	abc123	2019-03-28
2	2	test1	1	abc123	2019-03-28
3	3	test2	2	qwerty	2019-03-28
4	4	test3	3	abcdef	2019-03-29
5	5	test4	3	bbbbbb	2019-03-29

Report table:

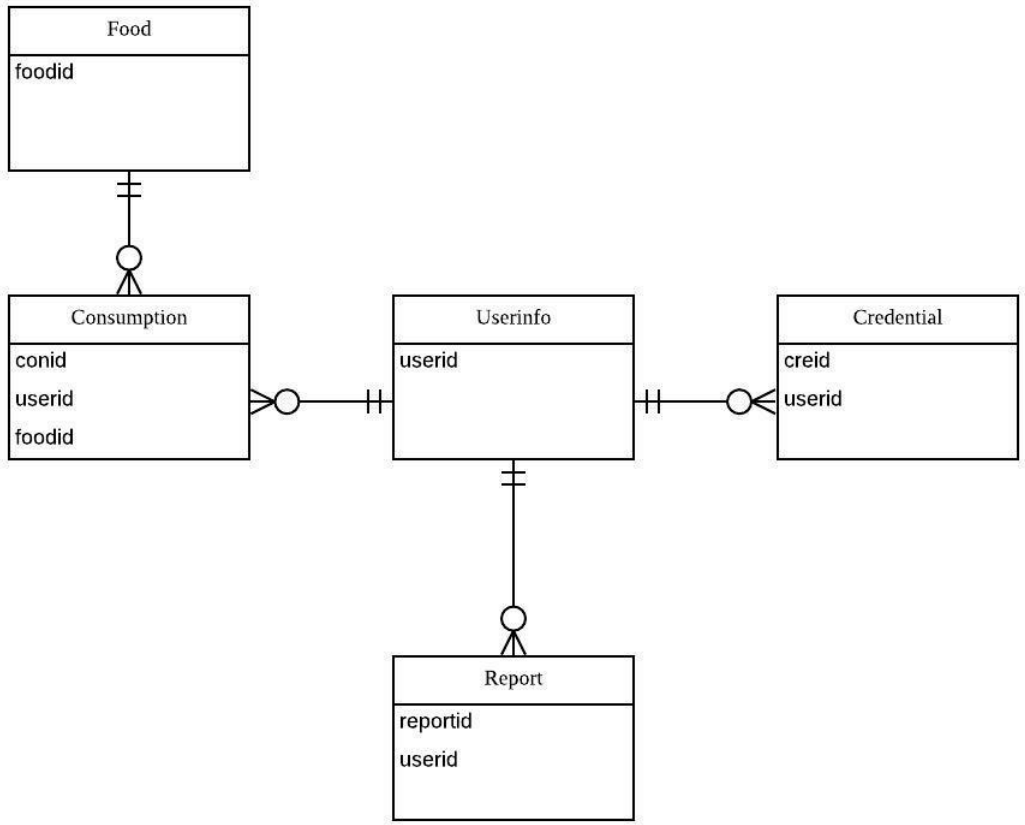
```
create table "WQW".REPORT
(
    REPORTID INTEGER not null primary key,
    USERID INTEGER not null,
    DATE DATE not null,
    TCALORIE_CONSUM INTEGER,
    TCALORIE_BURN INTEGER,
    TSTEPS INTEGER,
```

```
    CALORIE_GOAL INTEGER
)
ALTER TABLE REPORT
ADD CONSTRAINT FK_REPORT_USERID
FOREIGN KEY (USERID) REFERENCES USERINFO (USERID)
ON DELETE CASCADE;
```

Records in the report table:

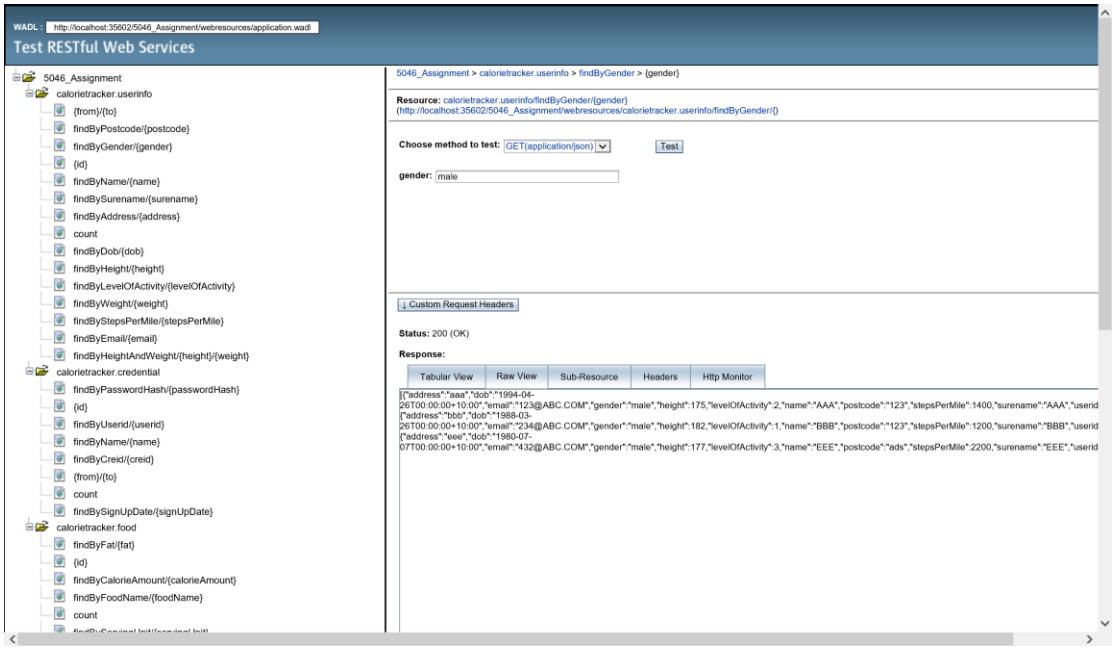
#	REPORTID	USERID	DATE	TCALORIE_CONSUM	TCALORIE_BURN	TSTEPS	CALORIE_GOAL
1	1	1	2019-03-29	400	260	3600	350
2	2	2	2019-03-29	200	400	6000	350
3	3	3	2019-03-29	300	300	4000	220
4	4	4	2019-03-29	280	430	7000	400
5	5	5	2019-03-29	320	100	1500	200
6	6	1	2019-04-01	400	300	1000	700

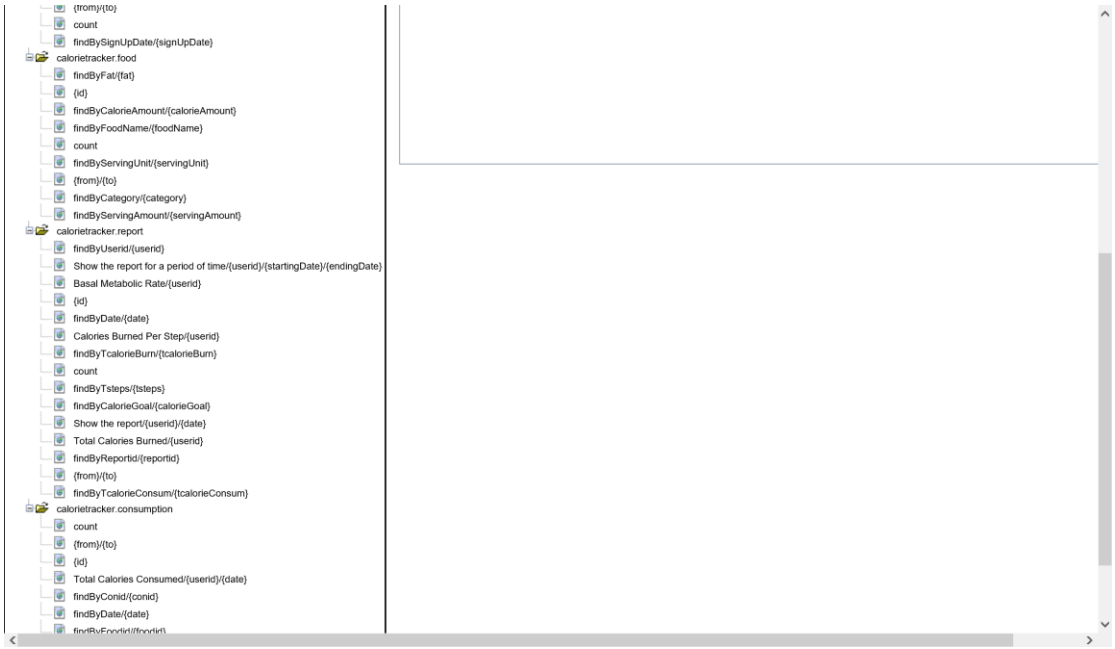
ER diagram of CalorieTracker DB:



Task 2

Test RESTful web service, all the methods are shown below:

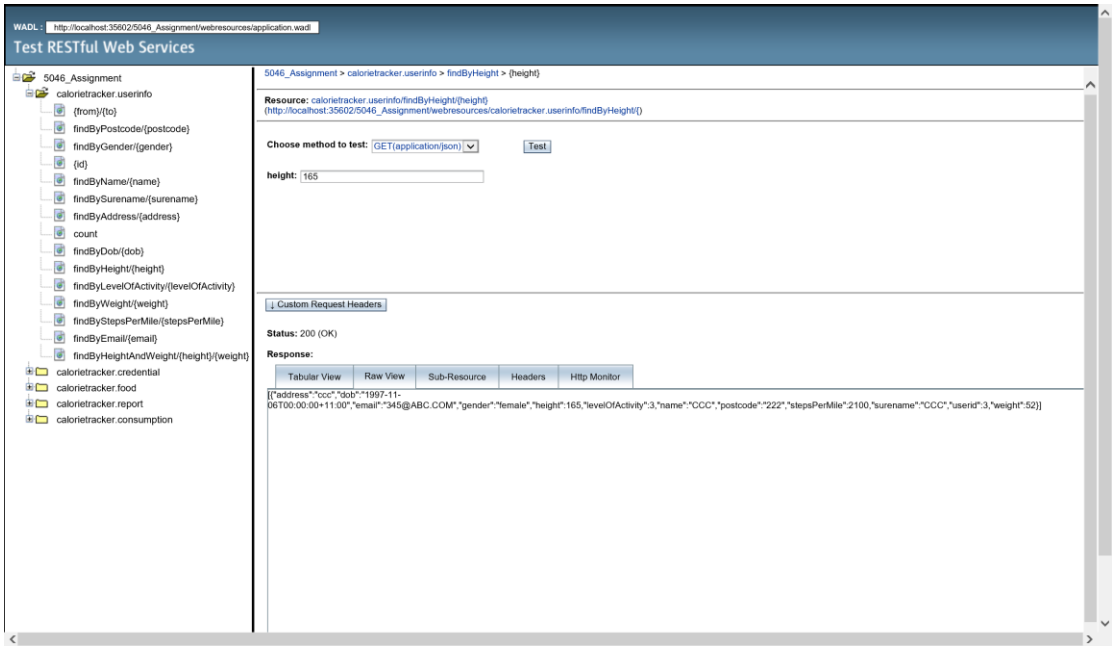




Task 3

NOTICE: The new methods I added will be shown and explained in the next two tasks

a) Methods of User table



Find by first name:

```
@GET
    @Path("findByName/{name}")
    @Produces({"application/json"})
    public List<Userinfo> findByName(@PathParam("name") String name) {
        Query query = em.createNamedQuery("Userinfo.findByName");
        query.setParameter("name", name);
        return query.getResultList();
    }
```

Find by sure name:

```
    @GET
    @Path("findBySurename/{surename}")
    @Produces({"application/json"})
    public List<Userinfo> findBySurename(@PathParam("surename") String surename) {
        Query query = em.createNamedQuery("Userinfo.findBySurename");
        query.setParameter("surename", surename);
        return query.getResultList();
```

```
    }
```

Find by email:

```
@GET
    @Path("findByEmail/{email}")
    @Produces({"application/json"})
    public List<Userinfo> findByEmail(@PathParam("email") String email) {
        Query query = em.createNamedQuery("Userinfo.findByEmail");
        query.setParameter("email", email);
        return query.getResultList();
    }
```

Find by date of birth:

```
    @GET
    @Path("findByDob/{dob}")
    @Produces({"application/json"})
    public List<Userinfo> findByDob(@PathParam("dob") Date dob) {
        Query query = em.createNamedQuery("Userinfo.findByDob");
        query.setParameter("dob", dob);
        return query.getResultList();
    }
```

Find by height:

```
    @GET
    @Path("findByHeight/{height}")
    @Produces({"application/json"})
    public List<Userinfo> findByHeight(@PathParam("height") Integer height) {
        Query query = em.createNamedQuery("Userinfo.findByHeight");
        query.setParameter("height", height);
        return query.getResultList();
    }
```

Find by weight:

```
    @GET
    @Path("findByWeight/{weight}")
    @Produces({"application/json"})
```

```
public List<Userinfo> findByWeight(@PathParam("weight") Integer weight) {
    Query query = em.createNamedQuery("Userinfo.findByWeight");
    query.setParameter("weight", weight);
    return query.getResultList();
}
```

Find by gender:

```
@GET
@Path("findByGender/{gender}")
@Produces({"application/json"})
public List<Userinfo> findByGender(@PathParam("gender") String gender) {
    Query query = em.createNamedQuery("Userinfo.findByGender");
    query.setParameter("gender", gender);
    return query.getResultList();
}
```

Find by address:

```
@GET
@Path("findByAddress/{address}")
@Produces({"application/json"})
public List<Userinfo> findByAddress(@PathParam("address") String address) {
    Query query = em.createNamedQuery("Userinfo.findByAddress");
    query.setParameter("address", address);
    return query.getResultList();
}
```

Find by postcode:

```
@GET
@Path("findByPostcode/{postcode}")
@Produces({"application/json"})
public List<Userinfo> findByPostcode(@PathParam("postcode") String postcode) {
    Query query = em.createNamedQuery("Userinfo.findByPostcode");
    query.setParameter("postcode", postcode);
    return query.getResultList();
}
```

Find by lvl of activity:

```
@GET
@Path("findByLevelOfActivity/{levelOfActivity}")
@Produces({"application/json"})
public List<Userinfo> findByLevelOfActivity(@PathParam("levelOfActivity") Integer levelOfActivity) {
    Query query = em.createNamedQuery("Userinfo.findByLevelOfActivity");
    query.setParameter("levelOfActivity", levelOfActivity);
    return query.getResultList();
}
```

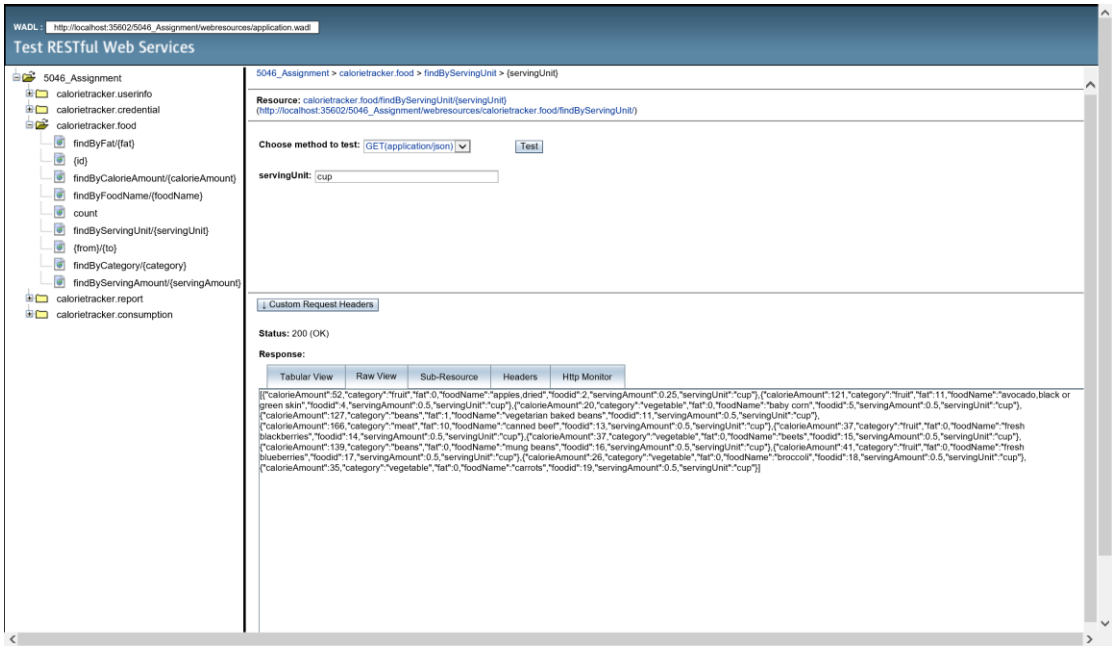
Find by steps per mile:

```
@GET
@Path("findByStepsPerMile/{stepsPerMile}")
@Produces({"application/json"})
public List<Userinfo> findByStepsPerMile(@PathParam("stepsPerMile") Integer stepsPerMile) {
    Query query = em.createNamedQuery("Userinfo.findByStepsPerMile");
    query.setParameter("stepsPerMile", stepsPerMile);
    return query.getResultList();
}
```

Find by user whose height and weight are greater than the input number:

```
@GET
@Path("findByHeightAndWeight/{height}/{weight}")
@Produces({"application/json"})
public List<Userinfo> findByHeightAndWeight(@PathParam("height") Integer height, @PathParam("weight") Integer weight) {
    TypedQuery<Userinfo> query = em.createQuery(
    "SELECT u FROM Userinfo u WHERE u.height >= :height AND u.weight >= :weight", Userinfo.class);
    query.setParameter("height", height);
    query.setParameter("weight", weight);
    return query.getResultList();
}
```

Methods of Food table



Find by food name:

```
@GET
   @Path("findByFoodName/{foodName}")
   @Produces({"application/json"})
   public List<Food> findByFoodName(@PathParam("foodName") String foodName) {
      Query query = em.createNamedQuery("Food.findByFoodName");
      query.setParameter("foodName", foodName);
      return query.getResultList();
   }
```

Find by food category:

```
   @GET
   @Path("findByCategory/{category}")
   @Produces({"application/json"})
   public List<Food> findByCategory(@PathParam("category") String category) {
      Query query = em.createNamedQuery("Food.findByCategory");
      query.setParameter("category", category);
      return query.getResultList();
   }
```

Find by calorie amount:

```
@GET
@Path("findByCalorieAmount/{calorieAmount}")
@Produces({"application/json"})
public List<Food> findByCalorieAmount(@PathParam("calorieAmount") Integer calorieAmount) {
    Query query = em.createNamedQuery("Food.findByCalorieAmount");
    query.setParameter("calorieAmount", calorieAmount);
    return query.getResultList();
}
```

Find by serving unit:

```
@GET
@Path("findByServingUnit/{servingUnit}")
@Produces({"application/json"})
public List<Food> findByServingUnit(@PathParam("servingUnit") String servingUnit) {
    Query query = em.createNamedQuery("Food.findByServingUnit");
    query.setParameter("servingUnit", servingUnit);
    return query.getResultList();
}
```

Find by serving amount:

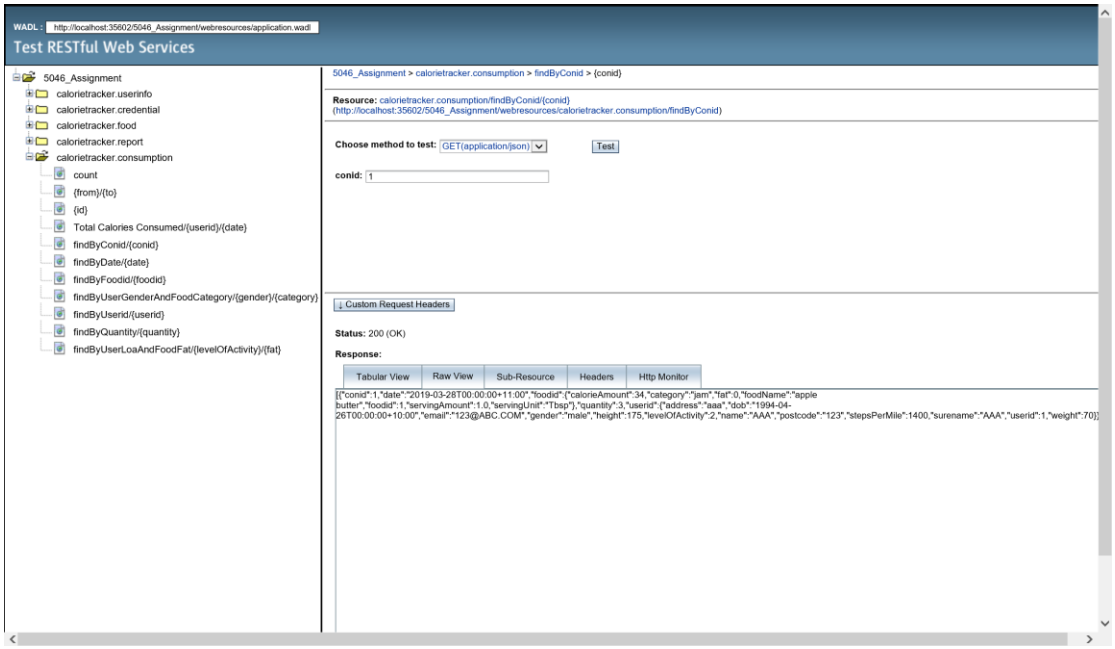
```
@GET
@Path("findByServingAmount/{servingAmount}")
@Produces({"application/json"})
public List<Food> findByServingAmount(@PathParam("servingAmount") Double servingAmount) {
    Query query = em.createNamedQuery("Food.findByServingAmount");
    query.setParameter("servingAmount", servingAmount);
    return query.getResultList();
}
```

Find by fat

```
@GET
@Path("findByFat/{fat}")
@Produces({"application/json"})
```

```
public List<Food> findByFat(@PathParam("fat") Integer fat) {
    Query query = em.createNamedQuery("Food.findByFat");
    query.setParameter("fat", fat);
    return query.getResultList();
}
```

Methods of Consumption table



Find by consumption id:

```
@GET
   @Path("findByConid/{conid}")
   @Produces({"application/json"})
   public List<Consumption> findByConid(@PathParam("conid") Integer conid) {
      Query query = em.createNamedQuery("Consumption.findByConid");
      query.setParameter("conid", conid);
      return query.getResultList();
   }
```

Find by date:

```
   @GET
   @Path("findByDate/{date}")
   @Produces({"application/json"})
   public List<Consumption> findByDate(@PathParam("date") Date date) {
      Query query = em.createNamedQuery("Consumption.findByDate");
      query.setParameter("date", date);
      return query.getResultList();
   }
```

Find by quantity:

```
@GET
@Path("findByQuantity/{quantity}")
@Produces({"application/json"})
public List<Consumption> findByQuantity(@PathParam("quantity") Integer quantity) {
    Query query = em.createNamedQuery("Consumption.findByQuantity");
    query.setParameter("quantity", quantity);
    return query.getResultList();
}
```

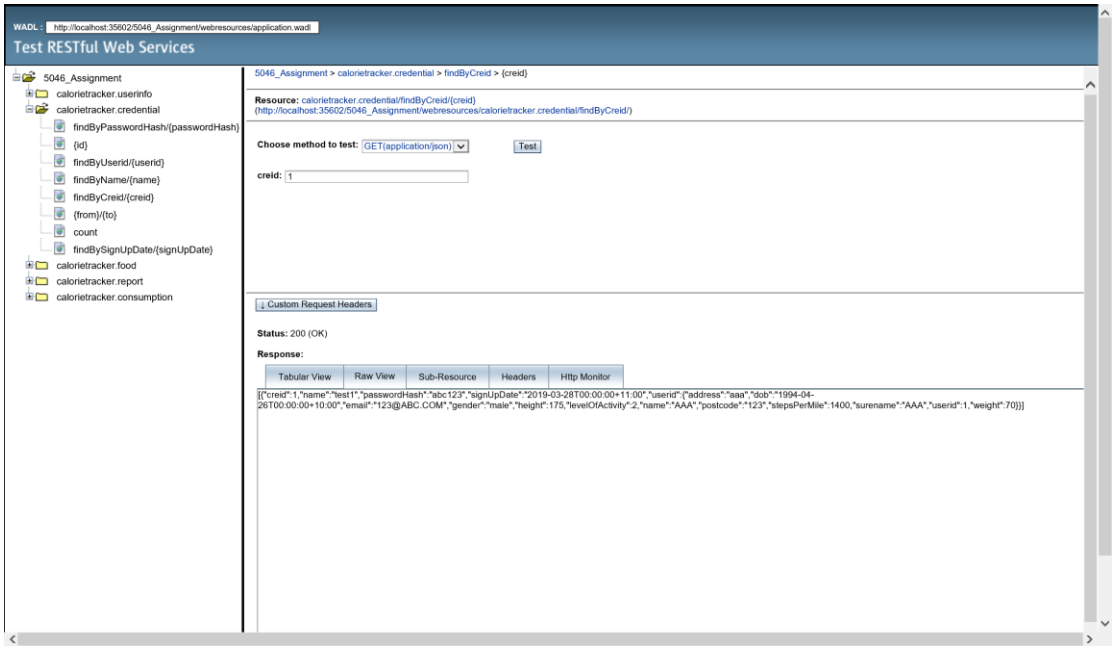
Find by user's gender and food's category:

```
@GET
@Path("findByUserGenderAndFoodCategory/{gender}/{category}")
@Produces({"application/json"})
public List<Consumption> findByUserGenderAndFoodCategory(@PathParam("gender") String gender, @PathParam("category") String category){
    TypedQuery<Consumption> q = em.createQuery("SELECT c FROM Consumption c WHERE c.userid.gender = :gender AND c.foodid.category = :category", Consumption.class);
    q.setParameter("gender", gender);
    q.setParameter("category", category);
    return q.getResultList();
}
@GET
@Path("findByUserLoaAndFoodFat/{levelOfActivity}/{fat}")
@Produces({"application/json"})
public List<Consumption> findByUserLoaAndFoodFat(@PathParam("levelOfActivity") Integer levelOfActivity, @PathParam("fat") Integer fat) {
    Query query = em.createNamedQuery("Consumption.findByUserLoaAndFoodFat");
    query.setParameter("levelOfActivity", levelOfActivity);
    query.setParameter("fat", fat);
    return query.getResultList();
}
@GET
@Path("findByFoodid/{foodid}")
```

```
    @Produces({"application/json"})
    public List<Consumption> findByFoodid(@PathParam("foodid") Integer foodid) {
        TypedQuery<Consumption> q = em.createQuery("SELECT c FROM Consumption c WHERE c.foodid.foodid = :foodid", Consumption.class);
        q.setParameter("foodid", foodid);
        return q.getResultList();
    }
    @GET
    @Path("findByUserid/{userid}")
    @Produces({"application/json"})
    public List<Consumption> findByUserid(@PathParam("userid") Integer userid) {
        TypedQuery<Consumption> q = em.createQuery("SELECT c FROM Consumption c WHERE c.userid.userid = :userid", Consumption.class);
        q.setParameter("userid", userid);
        return q.getResultList();
    }
    @GET
    @Path("Total Calories Consumed/{userid}/{date}")
    //@Produces({"application/json"})
    @Produces(MediaType.TEXT_PLAIN)
    public int totalCaloriesConsumed(@PathParam("userid") Integer userid, @PathParam("date") Date date) {
        int tcc = 0;
        TypedQuery<Consumption> q = em.createQuery("SELECT c FROM Consumption c WHERE c.userid.userid = :userid AND c.date = :date", Consumption.class);
        q.setParameter("userid", userid);
        q.setParameter("date", date);
        List<Consumption> con = q.getResultList();
        for(int i = 0;i < con.size();i ++){
            Food food = con.get(i).getFoodid();
            int calorie = food.getCalorieAmount() * con.get(i).getQuantity();
            tcc = tcc + calorie;
        }
        return tcc;
```

```
}
```

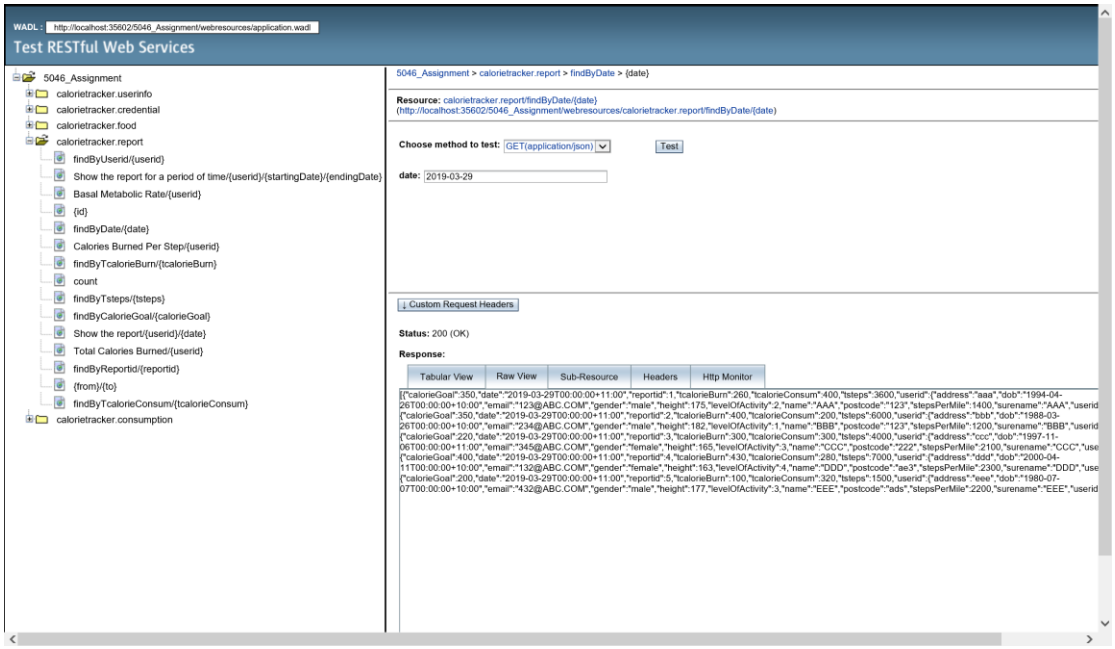
Methods of Credential table




```
@GET
    @Path("findByCreid/{creid}")
    @Produces({"application/json"})
    public List<Credential> findByCreid(@PathParam("creid") Integer creid) {
        Query query = em.createNamedQuery("Credential.findByCreid");
        query.setParameter("creid", creid);
        return query.getResultList();
    }
    @GET
    @Path("findByName/{name}")
    @Produces({"application/json"})
    public List<Credential> findByName(@PathParam("name") String name) {
        Query query = em.createNamedQuery("Credential.findByName");
        query.setParameter("name", name);
        return query.getResultList();
    }
    @GET
    @Path("findByPasswordHash/{passwordHash}")
    @Produces({"application/json"})
    public List<Credential> findByPasswordHash(@PathParam("passwordHash") String
```

```
passwordHash) {
        Query query = em.createNamedQuery("Credential.findByPasswordHash");
        query.setParameter("passwordHash", passwordHash);
        return query.getResultList();
    }
    @GET
    @Path("findBySignUpDate/{signUpDate}")
    @Produces({"application/json"})
    public List<Credential> findBySignUpDate(@PathParam("signUpDate") Date signUpDate)
{
        Query query = em.createNamedQuery("Credential.findBySignUpDate");
        query.setParameter("signUpDate", signUpDate);
        return query.getResultList();
    }
    @GET
    @Path("findByUserid/{userid}")
    @Produces({"application/json"})
    public List<Credential> findByUserid(@PathParam("userid") Integer userid) {
        TypedQuery<Credential> q = em.createQuery("SELECT c FROM Credential c WHERE
c.userid.userid = :userid", Credential.class);
        q.setParameter("userid", userid);
        return q.getResultList();
    }
```

Methods of Report table




```
@GET
    @Path("findByUserid/{userid}")
    @Produces({"application/json"})
    public List<Report> findByUserid(@PathParam("userid") Integer userid) {
        TypedQuery<Report> q = em.createQuery("SELECT r FROM Report r WHERE r.userid.userid = :userid", Report.class);
        q.setParameter("userid", userid);
        return q.getResultList();
    }
    @GET
    @Path("findByReportid/{reportid}")
    @Produces({"application/json"})
    public List<Report> findByReportid(@PathParam("reportid") Integer reportid) {
        Query query = em.createNamedQuery("Report.findByReportid");
        query.setParameter("reportid", reportid);
        return query.getResultList();
    }
    @GET
    @Path("findByDate/{date}")
    @Produces({"application/json"})
```

```
    public List<Report> findByDate(@PathParam("date") Date date) {
        Query query = em.createNamedQuery("Report.findByDate");
        query.setParameter("date", date);
        return query.getResultList();
    }
    @GET
    @Path("findByTcalorieConsum/{tcalorieConsum}")
    @Produces({"application/json"})
    public List<Report> findByTcalorieConsum(@PathParam("tcalorieConsum") Integer
tcalorieConsum) {
        Query query = em.createNamedQuery("Report.findByTcalorieConsum");
        query.setParameter("tcalorieConsum", tcalorieConsum);
        return query.getResultList();
    }
    @GET
    @Path("findByTcalorieBurn/{tcalorieBurn}")
    @Produces({"application/json"})
    public List<Report> findByTcalorieBurn(@PathParam("tcalorieBurn") Integer tcalorieBurn)
{
        Query query = em.createNamedQuery("Report.findByTcalorieBurn");
        query.setParameter("tcalorieBurn", tcalorieBurn);
        return query.getResultList();
    }
    @GET
    @Path("findByTsteps/{tsteps}")
    @Produces({"application/json"})
    public List<Report> findByTsteps(@PathParam("tsteps") Integer tsteps) {
        Query query = em.createNamedQuery("Report.findByTsteps");
        query.setParameter("tsteps", tsteps);
        return query.getResultList();
    }
    @GET
    @Path("findByCalorieGoal/{calorieGoal}")
```

```
@Produces({"application/json"})
public List<Report> findByCalorieGoal(@PathParam("calorieGoal") Integer calorieGoal) {
    Query query = em.createNamedQuery("Report.findByCalorieGoal");
    query.setParameter("calorieGoal", calorieGoal);
    return query.getResultList();
}
@GET
@Path("Calories Burned Per Step/{userid}")
@Produces(MediaType.TEXT_PLAIN)
public double caloriesBurnedPerStep(@PathParam("userid") Integer userid) {
    //Userinfo user = em.getReference(Userinfo.class, userid);
    Userinfo user = super.find(userid).getUserid();
    int weight = user.getWeight();
    int stepsPerMile = user.getStepsPerMile();
    double cbps = 0;
    cbps = (weight * 0.49 * 2.205) / stepsPerMile;
    return cbps;
}
@GET
@Path("Basal Metabolic Rate/{userid}")
@Produces(MediaType.TEXT_PLAIN)
public double basalMetabolicRate(@PathParam("userid") Integer userid) {
    Calendar cal = Calendar.getInstance();
    Calendar dob = Calendar.getInstance();
    Userinfo user = super.find(userid).getUserid();
    int age = 0;
    age = cal.get(Calendar.YEAR) - dob.get(Calendar.YEAR);
    if (cal.get(Calendar.DAY_OF_YEAR) > dob.get(Calendar.DAY_OF_YEAR)) {
        age += 1;
    }
    double bmr = 0;
    if ("male" == user.getGender()) {
```

```
        bmr = (13.75 * user.getWeight()) + (5.003 * user.getHeight() - (6.755 * age) + 66.5);
    } else {
        bmr = (9.563 * user.getWeight()) + (1.85 * user.getHeight() - (4.676 * age) + 655.1);
    }
    return bmr;
}
@GET
@Path("Total Calories Burned/{userid}")
@Produces(MediaType.TEXT_PLAIN)
public double totalCaloriesBurned(@PathParam("userid") Integer userid) {
    //Userinfo user = em.getReference(Userinfo.class, userid);
    Userinfo user = super.find(userid).getUserid();
    double tcb = 0;
    double bmr = basalMetabolicRate(userid);
    int loa = user.getLevelOfActivity();
    switch (loa) {
        case 1:
            tcb = bmr * 1.2;
            break;
        case 2:
            tcb = bmr * 1.375;
            break;
        case 3:
            tcb = bmr * 1.55;
            break;
        case 4:
            tcb = bmr * 1.725;
            break;
        case 5:
            tcb = bmr * 1.9;
            break;
    }
```

```
        return tcb;
    }
    @GET
    @Path("Show the report/{userid}/{date}")
    @Produces(MediaType.APPLICATION_JSON)
    public ShowReport showTheReport(@PathParam("userid") Integer userid, @PathParam("date") Date date) {
        TypedQuery<Report> q = em.createQuery("SELECT r FROM Report r WHERE r.userid.userid = :userid AND r.date = :date", Report.class);
        q.setParameter("userid", userid);
        q.setParameter("date", date);
        Report rep = q.getSingleResult();
        int calorieGoal = rep.getCalorieGoal();
        int tcalorieConsum = rep.getTcalorieConsum();
        int tcalorieBurn = rep.getTcalorieBurn();
        int remainedCalorie = 0;
        remainedCalorie = calorieGoal - tcalorieConsum + tcalorieBurn;
        if(remainedCalorie <= 0){
            remainedCalorie = 0;
        }
        ShowReport sr = new ShowReport(tcalorieConsum, tcalorieBurn, remainedCalorie);// tcalorieBurnedAtRest);
        return sr;
    }
    @GET
    @Path("Show the report for a period of time/{userid}/{startingDate}/{endingDate}")
    @Produces(MediaType.APPLICATION_JSON)
    public ShowPeriodReport showThePeriodReport(@PathParam("userid") Integer userid, @PathParam("startingDate") Date startingdate, @PathParam("endingDate") Date endingdate) {
        TypedQuery<Report> q = em.createQuery("SELECT r FROM Report r WHERE r.userid.userid = :userid AND r.date >= :startingdate AND r.date <= :endingdate", Report.class);
        q.setParameter("userid", userid);
        q.setParameter("startingdate", startingdate);
```

```
        q.setParameter("endingdate", endingdate);
        List<Report> rep = q.getResultList();
        int totalCalorieConsum = 0;
        int totalCalorieBurn = 0;
        int totalSteps = 0;
        for(int i = 0; i < rep.size(); i++){
            totalCalorieConsum = totalCalorieConsum + rep.get(i).getTcalorieConsum();
            totalCalorieBurn = totalCalorieBurn + rep.get(i).getTcalorieBurn();
            totalSteps = totalSteps + rep.get(i).getTsteps();
        }
        ShowPeriodReport spr = new ShowPeriodReport(totalCalorieConsum, totalCalorieBurn, totalSteps);
        return spr;
    }
```

b) Find users whose height and weight are greater than the inputs and return a list of users.

```
    @GET
        @Path("findByHeightAndWeight/{height}/{weight}")
        @Produces({"application/json"})
        public List<Userinfo> findByHeightAndWeight(@PathParam("height") Integer height, @PathParam("weight") Integer weight) {
            TypedQuery<Userinfo> query = em.createQuery(
            "SELECT u FROM Userinfo u WHERE u.height >= :height AND u.weight >= :weight", Userinfo.class);
            query.setParameter("height", height);
            query.setParameter("weight", weight);
            return query.getResultList();
        }
```

c) Find consumption records by user’s gender and food’s category.

```
    @GET
        @Path("findByUserGenderAndFoodCategory/{gender}/{category}")
        @Produces({"application/json"})
        public List<Consumption> findByUserGenderAndFoodCategory(@PathParam("gender")
```

String gender, @PathParam("category") String category){

TypedQuery<Consumption> q = em.createQuery("SELECT c FROM Consumption c WHERE c.userid.gender = :gender AND c.foodid.category = :category", Consumption.class);

q.setParameter("gender", gender);

q.setParameter("category", category);

return q.getResultList();

}

d) Find consumption records by user's lvl of activity and food's fat.

@GET

@Path("findByUserLoaAndFoodFat/{levelOfActivity}/{fat}")

@Produces({"application/json"})

public List<Consumption> findByUserLoaAndFoodFat(@PathParam("levelOfActivity") Integer levelOfActivity, @PathParam("fat") Integer fat) {

Query query = em.createNamedQuery("Consumption.findByUserLoaAndFoodFat");

query.setParameter("levelOfActivity", levelOfActivity);

query.setParameter("fat", fat);

return query.getResultList();

}

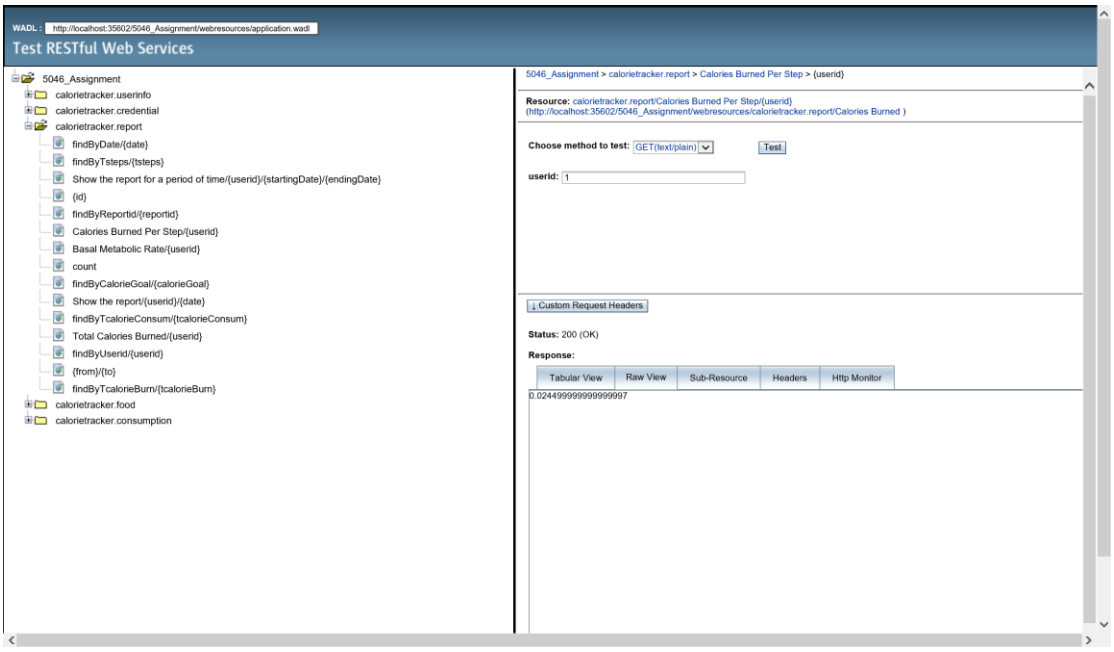
Namedquery of task d):

@NamedQuery(name = "Consumption.findByUserLoaAndFoodFat", query = "SELECT c FROM Consumption c WHERE c.userid.levelOfActivity = :levelOfActivity AND c.foodid.fat = :fat")})

Task 4

a) using find() method to get an user object and calculate user's calories burned per step

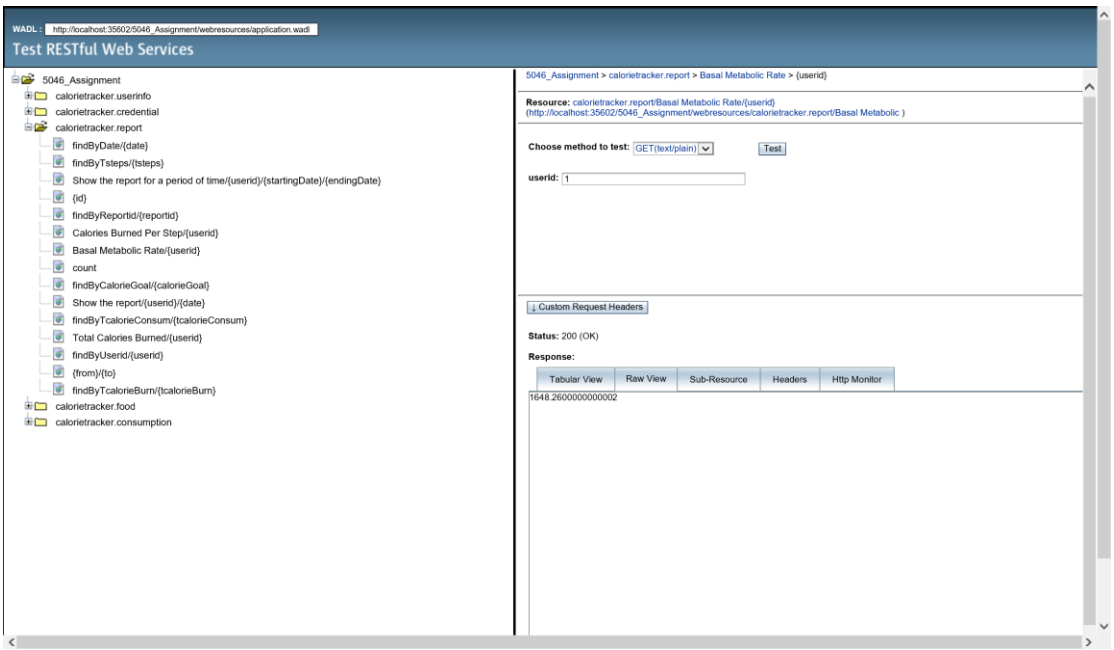
```
@GET
    @Path("Calories Burned Per Step/{userid}")
    @Produces(MediaType.TEXT_PLAIN)
    public double caloriesBurnedPerStep(@PathParam("userid") Integer userid) {
        Userinfo user = super.find(userid).getUserid();
        int weight = user.getWeight();
        int stepsPerMile = user.getStepsPerMile();
        double cbps = 0;
        cbps = (weight * 0.49 * 2.205) / stepsPerMile;
        return cbps;
    }
```



b) calculate user's BMR, I used Calendar class to get user's year of birth and compare to the current year to calculate the age.

```
@GET
    @Path("Basal Metabolic Rate/{userid}")
    @Produces(MediaType.TEXT_PLAIN)
    public double basalMetabolicRate(@PathParam("userid") Integer userid) {
```

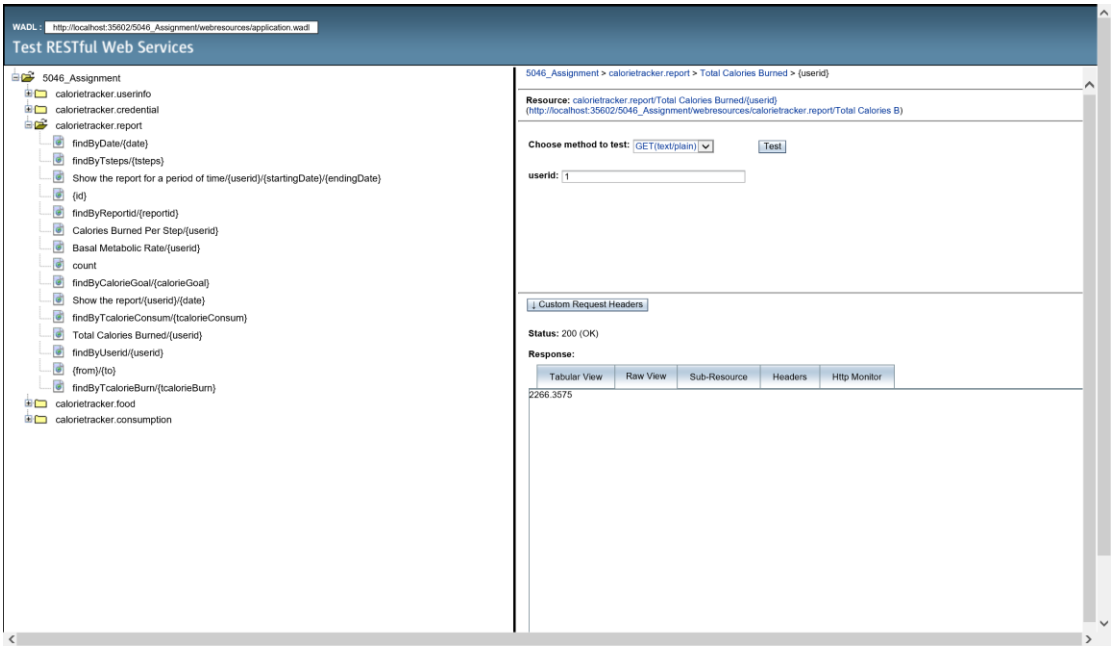
```
        Calendar cal = Calendar.getInstance();
        Calendar dob = Calendar.getInstance();
        //Userinfo user = em.getReference(Userinfo.class, userid);
        Userinfo user = super.find(userid).getUserid();
        int age = 0;
        age = cal.get(Calendar.YEAR) - dob.get(Calendar.YEAR);
        if (cal.get(Calendar.DAY_OF_YEAR) > dob.get(Calendar.DAY_OF_YEAR)) {
            age += 1;
        }
        double bmr = 0;
        if ("male" == user.getGender()) {
            bmr = (13.75 * user.getWeight()) + (5.003 * user.getHeight() - (6.755 * age) + 66.5);
        } else {
            bmr = (9.563 * user.getWeight()) + (1.85 * user.getHeight() - (4.676 * age) + 655.1);
        }
        return bmr;
    }
```



c) calculate user's calories burned based on BMR and lvl of activity used switch method

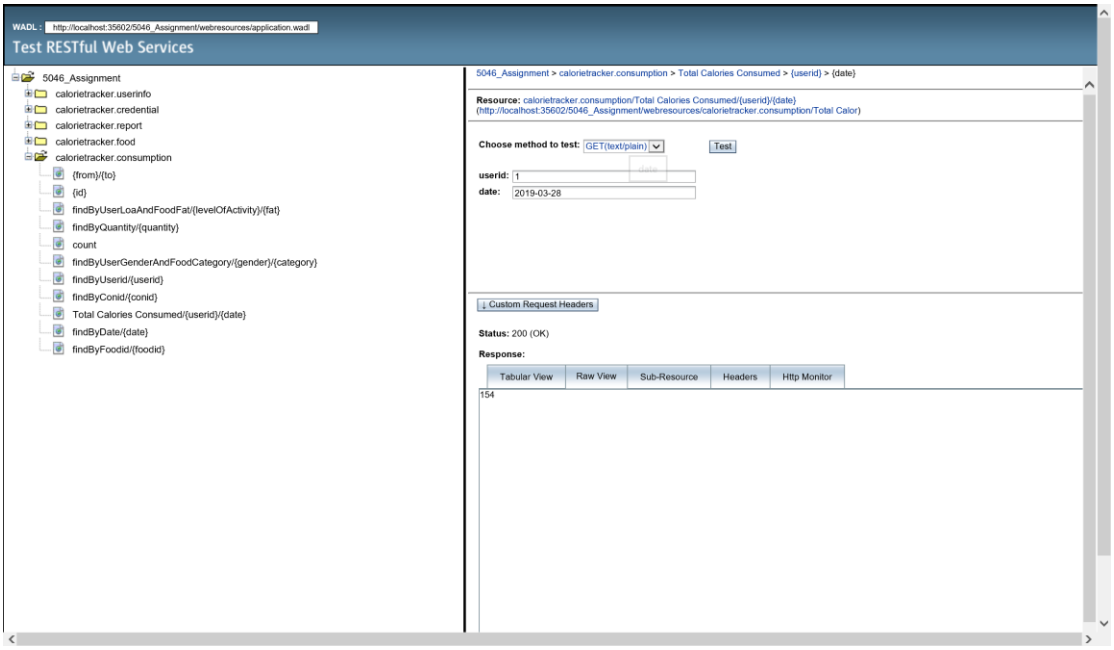
```
    @GET
    @Path("Total Calories Burned/{userid}")
```

```
@Produces(MediaType.TEXT_PLAIN)
public double totalCaloriesBurned(@PathParam("userid") Integer userid) {
    //Userinfo user = em.getReference(Userinfo.class, userid);
    Userinfo user = super.find(userid).getUserid();
    double tcb = 0;
    double bmr = basalMetabolicRate(userid);
    int loa = user.getLevelOfActivity();
    switch (loa) {
        case 1:
            tcb = bmr * 1.2;
            break;
        case 2:
            tcb = bmr * 1.375;
            break;
        case 3:
            tcb = bmr * 1.55;
            break;
        case 4:
            tcb = bmr * 1.725;
            break;
        case 5:
            tcb = bmr * 1.9;
            break;
    }
    return tcb;
}
```



d) calculate calories consumed based on userid and date

```
@GET
   @Path("Total Calories Consumed/{userid}/{date}")
   @Produces(MediaType.TEXT_PLAIN)
   public int totalCaloriesConsumed(@PathParam("userid") Integer userid, @PathParam("date") Date date) {
      int tcc = 0;
      TypedQuery<Consumption> q = em.createQuery("SELECT c FROM Consumption c WHERE c.userid.userid = :userid AND c.date = :date", Consumption.class);
      q.setParameter("userid", userid);
      q.setParameter("date", date);
      List<Consumption> con = q.getResultList();
      for(int i = 0;i < con.size();i ++){
         Food food = con.get(i).getFoodid();
         int calorie = food.getCalorieAmount() * con.get(i).getQuantity();
         tcc = tcc + calorie;
      }
      return tcc;
   }
```


WADL : http://localhost:35602/5046_Assignment/webresources/application.wadl
Test RESTful Web Services
5046_Assignment
calorietracker.userinfo
calorietracker.credential
calorietracker.report
calorietracker.food
calorietracker.consumption
{from}/{to}
{id}
findByUserLoaAndFoodFat/{levelOfActivity}/{fat}
findByQuantity/{quantity}
count
findByUserGenderAndFoodCategory/{gender}/{category}
findByUserId/{userid}
findByConid/{conid}
Total Calories Consumed/{userid}/{date}
findByDate/{date}
findByFoodid/{foodid}
5046_Assignment > calorietracker.consumption > Total Calories Consumed > {userid} > {date}
Resource: calorietracker.consumption/Total Calories Consumed/{userid}/{date}
(http://localhost:35602/5046_Assignment/webresources/calorietracker.consumption/Total Calor)
Choose method to test: GET(text/plain)
Test
userId: 1
date: 2019-03-28
↓ Custom Request Headers
Status: 200 (OK)
Response:
Tabular View
Raw View
Sub-Resource
Headers
Http Monitor
154

Task 5

a)

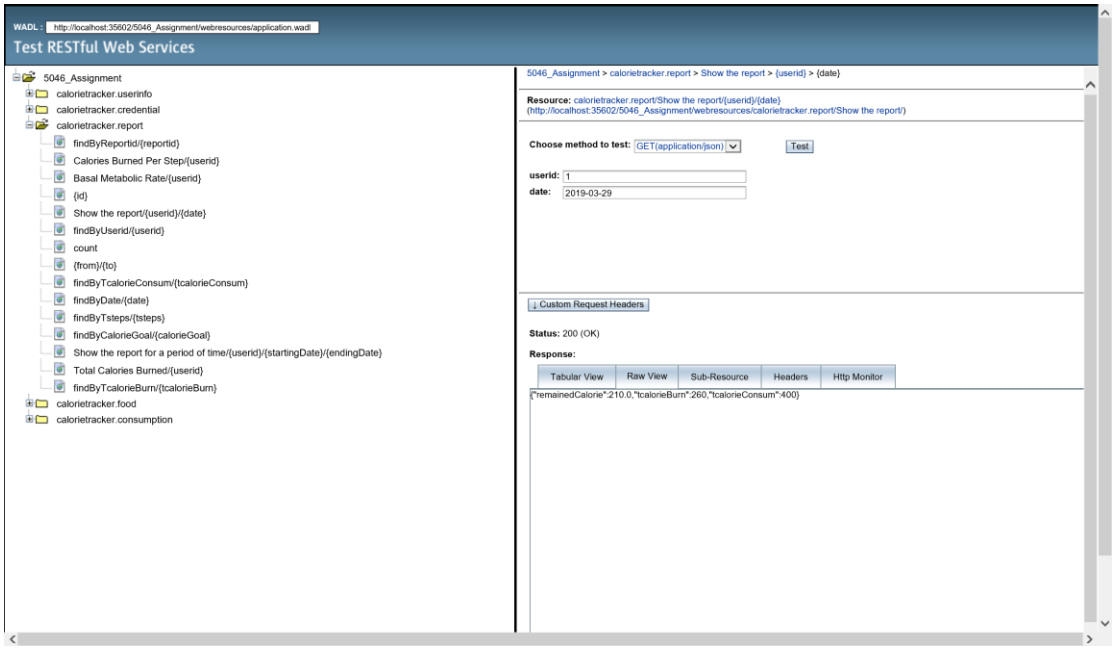
For this task, I added a new class called ShowReport, the code is shown below:

```
package CalorieReport;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement

public class ShowReport {

   int tcalorieConsum;
   int tcalorieBurn;
   double remainedCalorie;

   public int gettcalorieConsum() {
      return tcalorieConsum;
   }
   public int gettcalorieBurn() {
      return tcalorieBurn;
   }
   public double getremainedCalorie(){
      return remainedCalorie;
   }
   public void settcalorieConsum(int tcalorieConsum){
      this.tcalorieConsum = tcalorieConsum;
   }
   public void settcalorieBurn(int tcalorieBurn){
      this.tcalorieBurn = tcalorieBurn;
   }
   public void setremainedCalorie(double remainedCalorie){
      this.remainedCalorie = remainedCalorie;
```

```
}
public ShowReport(){
}
public ShowReport(int tcalorieConsum, int tcalorieBurn, double remainedCalorie){
    this.tcalorieConsum = tcalorieConsum;
    this.tcalorieBurn = tcalorieBurn;
    this.remainedCalorie = remainedCalorie;
}}
```

The method code in ReportREST class:

```
@GET
    @Path("Show the report/{userid}/{date}")
    @Produces(MediaType.APPLICATION_JSON)
    public ShowReport showTheReport(@PathParam("userid") Integer userid,
@PathParam("date") Date date) {
        TypedQuery<Report> q = em.createQuery("SELECT r FROM Report r WHERE
r.userid.userid = :userid AND r.date = :date", Report.class);
        q.setParameter("userid", userid);
        q.setParameter("date", date);
        Report rep = q.getSingleResult();
        int calorieGoal = rep.getCalorieGoal();
        int tcalorieConsum = rep.getTcalorieConsum();
        int tcalorieBurn = rep.getTcalorieBurn();
        int remainedCalorie = 0;
        remainedCalorie = calorieGoal - tcalorieConsum + tcalorieBurn;
        if(remainedCalorie <= 0){
            remainedCalorie = 0;
        }
        ShowReport sr = new ShowReport(tcalorieConsum, tcalorieBurn, remainedCalorie);//
tcalorieBurnedAtRest);
        return sr;
    }
```



b)

For this task, I added a new class called ShowPeriodReport, the code is shown below:

```
package CalorieReport;


import javax.xml.bind.annotation.XmlRootElement;


@XmlRootElement


public class ShowPeriodReport {


    int totalCalorieConsum;
    int totalCalorieBurn;
    int totalSteps;


    public int getConsum() {
        return totalCalorieConsum;
    }
    public int getBurn() {
        return totalCalorieBurn;
    }
```

```
    public int getSteps() {
        return totalSteps;
    }
    public void setConsum(int consum) {
        this.totalCalorieConsum = consum;
    }
    public void setBurn(int burn) {
        this.totalCalorieBurn = burn;
    }
    public void setSteps(int steps) {
        this.totalSteps = steps;
    }
    public ShowPeriodReport() {
    }
    public ShowPeriodReport(int consum, int burn, int steps) {
        this.totalCalorieBurn = burn;
        this.totalCalorieConsum = consum;
        this.totalSteps = steps;
    }
}
```

The method code:

```
@GET
    @Path("Show the report for a period of time/{userid}/{startingDate}/{endingDate}")
    @Produces(MediaType.APPLICATION_JSON)
    public ShowPeriodReport showThePeriodReport(@PathParam("userid") Integer userid, @PathParam("startingDate") Date startingdate, @PathParam("endingDate") Date endingdate) {
        TypedQuery<Report> q = em.createQuery("SELECT r FROM Report r WHERE r.userid.userid = :userid AND r.date >= :startingdate AND r.date <= :endingdate", Report.class);
        q.setParameter("userid", userid);
        q.setParameter("startingdate", startingdate);
        q.setParameter("endingdate", endingdate);
```

```
        List<Report> rep = q.getResultList();
        int totalCalorieConsum = 0;
        int totalCalorieBurn = 0;
        int totalSteps = 0;
        for(int i = 0; i < rep.size(); i++){
           totalCalorieConsum = totalCalorieConsum + rep.get(i).getTcalorieConsum();
           totalCalorieBurn = totalCalorieBurn + rep.get(i).getTcalorieBurn();
           totalSteps = totalSteps + rep.get(i).getTsteps();
        }
        ShowPeriodReport spr = new ShowPeriodReport(totalCalorieConsum,
totalCalorieBurn, totalSteps);
        return spr;
    }
```