Project 4 Task2 Writeup- BART Web Service

Name: Quinn Tian
Email: kunt@andrew.cmu.edu

URL of Web Server

https://still-fjord-16808.herokuapp.com/
https://still-fjord-16808.herokuapp.com/dashboard
https://still-fjord-16808.herokuapp.com/trainSchedule

Documentation for how the project requirements are met in my work:

1. Log useful information

At least 6 pieces of information is logged for each request/reply with the mobile phone. It should include information about the request from the mobile phone, information about the request and reply to the 3rd party API, and information about the reply to the mobile phone. (You should NOT log data from interactions from the operations dashboard.)

The below 7 pieces of information is saved in class Log and logged into MongoDB collections 'doc'. The information from request from mobile is saved as 'input'; request to 3p Api is always the same Url, so no need to save; reply status from Api is saved as 'from3pApi'; reply to mobile is saved as 'reply'.

```
Log log=new Log(timestamp, date, trainNo, station, input, from3pApi, reply);

bartModel.addToMongo(log); //call method to save Log into DB
```

2. Store the log information in a database

The web service can connect, store, and retrieve information from a MongoDB database in the cloud.

The addToMongo method in BARTModel handles storing data.

```
public void addToMongo(Log log){
    try (MongoClient mongoClient = MongoClients.create(

"mongodb+srv://quinntian:Pang0902@cluster0.uu6lc.mongodb.net/BART?retryWrites=true&w=majority")){
        System.out.println("Connect to MongoDB");
        MongoDatabase database = mongoClient.getDatabase("BART"); //create database
        MongoCollection<Document> trainSchedule = database.getCollection("doc");//create the collection
        Gson gson=new Gson();
        Document doc=new Document("doc",gson.toJson(log));
        trainSchedule.insertOne(doc);

        System.out.println("doc is inserted to MongoDB");
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
```

The analytics() method in BARTModel handles retrieve data from MongoDB and do analytics work.

```
public  List<Log> logs=new ArrayList<Log>();
public double api200Pct;
public  int totalLogs=0;
public int nullInputs=0;
public int api200=0;
public TreeMap<String, Integer> trainCounts=new TreeMap<String, Integer>();
public String mostPopularTrain;
int maxTrainCount;
public void analytics(){
    try (MongoClient mongoClient = MongoClients.create(

"mongodb+srv://quinntian:Pang0902@cluster0.uu6lc.mongodb.net/BART?retryWrites=true&w=majority")){
        System.out.println("Connect to MongoDB");
        MongoDatabase database = mongoClient.getDatabase("BART"); //create database
        MongoCollection<Document> trainSchedule = database.getCollection("doc");//create the collection
```

```
        Gson gson=new Gson();
        FindIterable<Document> iterDoc = trainSchedule.find();
        Iterator it = iterDoc.iterator();
        //MongoCursor<Document> cursor=iterDoc.iterator(); //another way to loop through collection
       /* while (cursor.hasNext()) {
            System.out.println(cursor.next());
        }*/

        //Citation: https://www.tutorialspoint.com/how-to-retrieve-all-the-documents-from-a-mongodb-
collection-using-java
        try{
            while (it.hasNext()) {
                Document d1= (Document) it.next();
                String s= (String) d1.get("doc");
                //Gson gson=new Gson();
                Log log=gson.fromJson(s, Log.class);
                logs.add(log);
                System.out.println(log.timestamp);

                totalLogs++;
                if (log.input==null) nullInputs++;
                if (log.from3pApi!=null && log.from3pApi.equals("200")) api200++;
                System.out.println("line 194 "+log.trainNo);
                if (log.trainNo!=null && !trainCounts.containsKey(log.trainNo)) {
                    //trainList.add(log.trainNo);
                    trainCounts.put(log.trainNo, 1);
                }
                if (log.trainNo!=null && trainCounts.containsKey(log.trainNo)) {
                    trainCounts.put(log.trainNo, (trainCounts.get(log.trainNo) + 1));
                }
            }
            for (String train: trainCounts.keySet()){
                if (trainCounts.get(train)>maxTrainCount) {
                    maxTrainCount=trainCounts.get(train);
                    mostPopularTrain=train;
                }
            }
            System.out.println("logs size "+logs.size());
            //System.out.println("log0 "+ logs.get(0).timestamp);
            api200Pct=(double) Math.round(api200*100/totalLogs*10.0)/10.0;
```

```
            System.out.println(api200+" out of "+totalLogs+" returned data from Api");
            System.out.println("Total of null input is "+nullInputs);
            System.out.println(api200Pct+"% of total have 200 status from 3p Api");
            System.out.println("The most popular train is " + mostPopularTrain+ ", with "
            + maxTrainCount + " times searched." );
        }
        finally {
            mongoClient.close();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

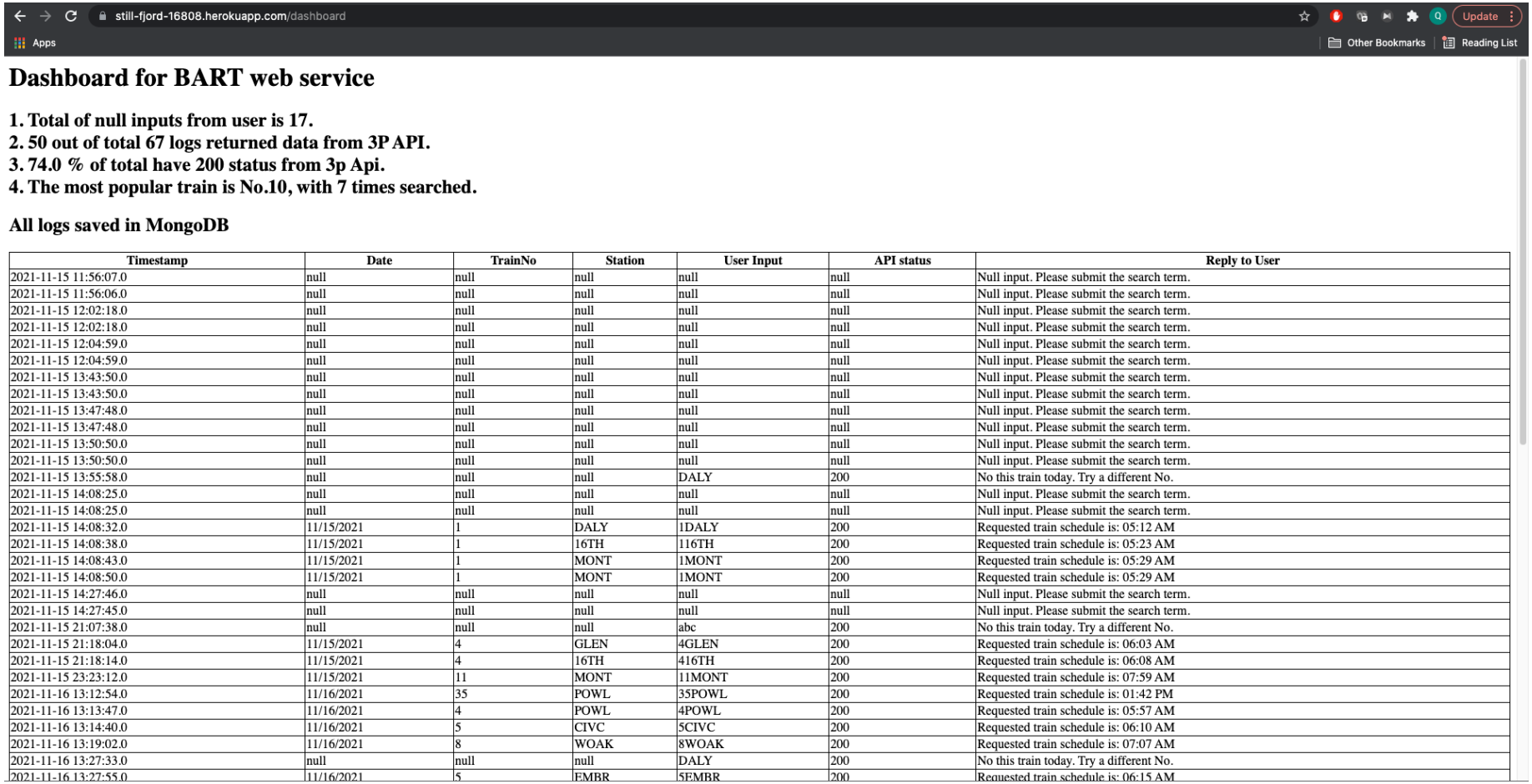
The result is passed to dashboard.jsp through BARTServlet 'doGet' method.

```
//url: /dashboard would present analytics and log history
if (request.getRequestURL().toString().contains("dashboard")) {
    PrintWriter out=response.getWriter();
    String url = request.getRequestURI();
    System.out.println("Request url :"+url);
    bartModel.analytics();
    request.setAttribute("totalLogs", bartModel.totalLogs);
    request.setAttribute("nullInputs", bartModel.nullInputs);
    request.setAttribute("api200", bartModel.api200);
    request.setAttribute("api200Pct", bartModel.api200Pct);
    request.setAttribute("trainNo", bartModel.mostPopularTrain);
    request.setAttribute("maxTrainCount", bartModel.maxTrainCount);

    request.setAttribute("logs", bartModel.logs);
    String nextView="dashboard.jsp";
    RequestDispatcher view = request.getRequestDispatcher(nextView);
    view.forward(request, response); //give view with request, response

}
```

3. Display operations analytics and full logs on a web-based dashboard



Dashboard for BART web service

1. Total of null inputs from user is 17.
2. 50 out of total 67 logs returned data from 3P API.
3. 74.0 % of total have 200 status from 3p Api.
4. The most popular train is No.10, with 7 times searched.

All logs saved in MongoDB

Timestamp	Date	TrainNo	Station	User Input	API status	Reply to User
2021-11-15 11:56:07.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 11:56:06.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 12:02:18.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 12:02:18.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 12:04:59.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 12:04:59.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 13:43:50.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 13:43:50.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 13:47:48.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 13:47:48.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 13:50:50.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 13:50:50.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 13:55:58.0	null	null	null	DALY	200	No this train today. Try a different No.
2021-11-15 14:08:25.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 14:08:25.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 14:08:32.0	11/15/2021	1	DALY	1DALY	200	Requested train schedule is: 05:12 AM
2021-11-15 14:08:38.0	11/15/2021	1	16TH	116TH	200	Requested train schedule is: 05:23 AM
2021-11-15 14:08:43.0	11/15/2021	1	MONT	1MONT	200	Requested train schedule is: 05:29 AM
2021-11-15 14:08:50.0	11/15/2021	1	MONT	1MONT	200	Requested train schedule is: 05:29 AM
2021-11-15 14:27:46.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 14:27:45.0	null	null	null	null	null	Null input. Please submit the search term.
2021-11-15 21:07:38.0	null	null	null	abc	200	No this train today. Try a different No.
2021-11-15 21:18:04.0	11/15/2021	4	GLEN	4GLEN	200	Requested train schedule is: 06:03 AM
2021-11-15 21:18:14.0	11/15/2021	4	16TH	416TH	200	Requested train schedule is: 06:08 AM
2021-11-15 23:23:12.0	11/15/2021	11	MONT	11MONT	200	Requested train schedule is: 07:59 AM
2021-11-16 13:12:54.0	11/16/2021	35	POWL	35POWL	200	Requested train schedule is: 01:42 PM
2021-11-16 13:13:47.0	11/16/2021	4	POWL	4POWL	200	Requested train schedule is: 05:57 AM
2021-11-16 13:14:40.0	11/16/2021	5	CIVC	5CIVC	200	Requested train schedule is: 06:10 AM
2021-11-16 13:19:02.0	11/16/2021	8	WOAK	8WOAK	200	Requested train schedule is: 07:07 AM
2021-11-16 13:27:33.0	null	null	null	DALY	200	No this train today. Try a different No.
2021-11-16 13:27:55.0	11/16/2021	5	EMBR	5EMBR	200	Requested train schedule is: 06:15 AM

a. A unique URL addresses a web interface dashboard for the web service.
https://still-fjord-16808.herokuapp.com/dashboard

b. The dashboard displays at least 3 interesting operations analytics. (see screenshot above and dashboard.jsp file in below)
c. The dashboard displays **formatted** full logs. (see screenshot)

4. Deploy the web service to Heroku

This web service should have all the functionality of Task 1 but with the additional logging, database, and dashboard analytics functions.
In your Task 2 writeup be sure to include the dashboard URL!

Class Result for Task 1 and 2

```
package edu.cmu.kunt.bart2;


public class Result {
    private int responseCode;
    private String responseText;

    public int getResponseCode() { return responseCode; }
    public void setResponseCode(int code) { responseCode = code; }
    public String getResponseText() { return responseText; }
    public void setResponseText(String msg) { responseText = msg; }

    public String toString() { return responseCode + ":" + responseText; }
}
```

Class TrainSchedule for task 1 and 2

```
package ds.project4task1;

public class TrainSchedule {
    //String url;
    String trainNo;
    String date;
    String station;
    String origTime;

    public TrainSchedule( String index, String date, String station, String origTime) {
        //this.url=urlString;
        this.trainNo=index;
        this.date=date;
        this.station=station;
        this.origTime=origTime;
```

```
    }
}
```

Android Model Code for Task 1 and 2

```
package ds.cmu.edu.bart;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ConnectException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;

import java.net.URL;

import android.os.AsyncTask;


/*
 * This class provides capabilities to search for an image on Flickr.com given a search term.  The method
"search" is the entry to the class.
 * Network operations cannot be done from the UI thread, therefore this class makes use of an AsyncTask
inner class that will do the network
 * operations in a separate worker thread.  However, any UI updates should be done in the UI thread so
avoid any synchronization problems.
 * onPostExecution runs in the UI thread, and it calls the ImageView pictureReady method to do the update.
 *
 */
public class BartModel {
    BartController ip = null;

    /*
     * search is the public GetPicture method.  Its arguments are the search term, and the
InterestingPicture object that called it.  This provides a callback
```

```
     * path such that the pictureReady method in that object is called when the picture is available from
the search.
     */
    public void search(String searchTerm, BartController ip) {
        this.ip = ip;
        new AsyncCallAPI().execute(searchTerm);
    }
    public class AsyncCallAPI extends AsyncTask<String, String, String> {

        public AsyncCallAPI(){
            //set context variables if required
        }
        @Override
        //citation: https://stackoverflow.com/questions/2938502/sending-post-data-in-android
        protected String doInBackground(String... urls) { //params ??
            return getSchedule(urls[0]);
        }

        protected void onPostExecute(String response) {
            ip.scheduleReady(response);
        }

        private String getSchedule(String searchTerm)  {
            String response = "";


            try {
                //call on Heroku web service
                //URL url=new URL("https://calm-savannah-55275.herokuapp.com/trainSchedule/"//for task1
                //for task2
                //URL url=new URL("https://obscure-taiga-11818.herokuapp.com/trainSchedule/"
                URL url = new URL( "http://192.168.0.5:8080/Bart2-1.0-SNAPSHOT/trainSchedule/"
                        +searchTerm); //for local test
                System.out.println("URL = "+ url);
                System.out.println("Search term = " + searchTerm);
                /*
                 * Create an HttpURLConnection.  This is useful for setting headers
                 * and for getting the path of the resource that is returned (which
                 * may be different than the URL above if redirected).
                 * HttpsURLConnection (with an "s") can be used if required by the site.
                 */
```

```
                HttpURLConnection connection = (HttpURLConnection) url.openConnection();


                connection.setRequestMethod("GET"); //added!

                BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream(),
"UTF-8"));
                String str;
                // Read each line of "in" until done, adding each to "response"
                while ((str = in.readLine()) != null) {
                    // str is one line of text readLine() strips newline characters
                    response += str;
                }
                in.close();

            }
            catch (ConnectException e) {
                System.out.println("url connection failed");


            }catch (ProtocolException e ) {
                e.printStackTrace();

            } catch (MalformedURLException e){
                e.printStackTrace();

            } catch (IOException e ){
                e.printStackTrace();

            }
            return response;
        }
    }

}
```

Android Controller

```
package ds.cmu.edu.bart;
```

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioGroup;
import android.widget.TextView;
import com.google.gson.Gson;
import android.widget.RadioButton;
import ds.cmu.edu.interestingpicture.R;

public class BartController extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        /*
         * The click listener will need a reference to this object, so that upon successfully finding a
picture from Flickr, it
         * can callback to this object with the resulting picture Bitmap.  The "this" of the OnClick will
be the OnClickListener, not
         * this InterestingPicture.
         */
        final BartController ma = this;

        /*
         * Find the "submit" button, and add a listener to it
         */
        Button submitButton = (Button)findViewById(R.id.submit);
        RadioGroup radioNoGroup=(RadioGroup) findViewById(R.id.radioNo);


        // Add a listener to the send button
        submitButton.setOnClickListener(new View.OnClickListener(){
            public void onClick(View viewParam) {
                String trainNO = ((EditText)findViewById(R.id.searchTerm)).getText().toString();

                // get selected radio button from radioGroup
                int selected = radioNoGroup.getCheckedRadioButtonId();
```

```
                // find the radiobutton by returned id
                RadioButton radioNoButton = (RadioButton) findViewById(selected);
                String station=radioNoButton.getText().toString();
                String searchTerm=trainNO+station; //search term is concatenation

                System.out.println("searchTerm = " + searchTerm);

                BartModel gp = new BartModel();
                gp.search(searchTerm, ma); // Done asynchronously in another thread.  It calls
ip.pictureReady() in this thread when complete.
            }
        });
    }

    /*
     * This is called by the GetPicture object when the response from API is ready.  This allows for
passing back the Bitmap picture for updating the ImageView
     */

    public void scheduleReady(String response){
        TextView searchView = (EditText)findViewById(R.id.searchTerm);//read the textbox
        TextView responseView = findViewById(R.id.response);
        if (response=="") { //address the case not connecting to server
            responseView.setText("No response from Server. ");
            responseView.setVisibility(View.VISIBLE);
        }
        //if (response==null) responseView.setText("Sorry no schedule is available for your input");
        else {
            Gson gson=new Gson();
            //convert Json response to regular String
            String schedule=gson.fromJson(response,  String.class);
            //responseView.setText("Departure time: " +schedule.origTime);
            responseView.setText(schedule);
            responseView.setVisibility(View.VISIBLE);
            System.out.println( schedule);
        }
        //searchView.setText("");
        responseView.invalidate();

    }
```

```
}
```

Task 2 Server Servlet code

```
package edu.cmu.kunt.bart2;

/*
 * @author Quinn Tian
 * this is for Project4Task2
 */

import com.google.gson.Gson;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Timestamp;

@WebServlet(name = "BARTServlet",
        urlPatterns = {"/trainSchedule/*" ,"/dashboard"})
public class BARTServlet extends HttpServlet {

    BARTModel bartModel = null;  // The "business model" for this app

    // Initiate this servlet by instantiating the model that it will use.
    @Override
    public void init() {
        bartModel = new BARTModel ();
    }

    // This servlet will reply to HTTP GET requests via this doGet method
    @Override
```

```java
// Make an HTTP GET request
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    //if the url has /trainSchedule/* would get schedule for Android or web users
    //and save the data in MongoDB
    if (request.getRequestURL().toString().contains("trainSchedule")) {
        Timestamp timestamp=new Timestamp(System.currentTimeMillis());
        TrainSchedule returnedSchedule=null;
        String date=null; //train date is also current date
        String trainNo=null;
        String station=null;
        String from3pApi=null; //status from 3p Api

        String input=null; //input form user
        Gson gson=new Gson();
        String reply = null; //reply to user
        String jsonReply = "";
        PrintWriter out=response.getWriter();

        String url = request.getRequestURI();
        System.out.println("Request url from Android:"+url);

        if (url.contains("trainSchedule/")){
            String[] inputs=url.split("Schedule/");
            if (inputs.length>1) {
                input=inputs[1];
                System.out.println("Android input: "+input);
            }
        }
        //1.taking care of invalid mobile app input
        if (input==null || input.isEmpty()) {
            reply="Null input. Please submit the search term.";
        }
        if (input!=null && !input.isEmpty() ){ //get the schedule if available
            from3pApi=String.valueOf(BARTModel.fetchData().getResponseCode());
            returnedSchedule=BARTModel.getSchedule(input);
            if (returnedSchedule==null){ //taking care of the case that schedule is not available
                reply="No this train today. Try a different No.";
            }
            else { //read into TrainSchedule for each variable that will be saved in log
                date=returnedSchedule.date;
```

```
                    trainNo=returnedSchedule.trainNo;
                    station=returnedSchedule.station;
                    //this is reply message to user with retrieved schedule time
                    reply="Requested train schedule is: "+returnedSchedule.origTime;
                }
            }
            jsonReply=gson.toJson(reply);
            System.out.println("Json reply to Android: "+jsonReply);
            response.setContentType("application/json");
            response.setCharacterEncoding("UTF-8");
            out.print(jsonReply); //send reply to user
            out.close();
            //create a new Log to save into DB
            Log log=new Log(timestamp, date, trainNo, station, input, from3pApi, reply);

            bartModel.addToMongo(log); //call method to save Log into DB
        }
        //if url is with /dashboard, present analytics and log history to webpage via View file
        if (request.getRequestURL().toString().contains("dashboard")) {
            PrintWriter out=response.getWriter();
            String url = request.getRequestURI();
            System.out.println("Request url :"+url);
            bartModel.analytics();  //get all analytics results and load into bartModel
            //set dashboard attributes with all values from analytics results
            request.setAttribute("totalLogs", bartModel.totalLogs);
            request.setAttribute("nullInputs", bartModel.nullInputs);
            request.setAttribute("api200", bartModel.api200);
            request.setAttribute("api200Pct", bartModel.api200Pct);
            request.setAttribute("trainNo", bartModel.mostPopularTrain);
            request.setAttribute("maxTrainCount", bartModel.maxTrainCount);

            request.setAttribute("logs", bartModel.logs); //send all logs to View
            String nextView="dashboard.jsp";
            RequestDispatcher view = request.getRequestDispatcher(nextView);
            view.forward(request, response); //give view with request, response

        }
    }

}
```

Task2 Server Model code

```
package edu.cmu.kunt.bart2;
/*
        * @author Quinn Tian
        * this is for Project4Task2
        */
import com.google.gson.Gson;
import com.mongodb.client.*;

import org.bson.Document;

import org.json.JSONArray;
import org.json.JSONObject;
import org.json.JSONTokener;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.*;


public class BARTModel {
    public static Map<String, TrainSchedule> scheduleMap=new HashMap<>();
    //URL of 3p API
    static String urlString="https://api.bart.gov/api/sched.aspx?cmd=routesched&route=12&key=MW9S-E7SL-
26DU-VV8V&json=y";
    //load scheduleMap with String response received from 3p Api
    public static void loadScheduleMap(String response){

        //use org.json library to extract info from Json string extracted from 3p Api
        JSONTokener token = new JSONTokener(response); //split response
        JSONObject ob = new JSONObject(token);

        JSONObject root = (JSONObject)ob.get("root");
        String date=(String)root.get("date");
        System.out.println("Json root from Api: "+root);
```

```
        JSONObject route = (JSONObject)root.get("route");
        JSONArray trainArray=(JSONArray)route.get("train");
        for (int i=0; i<trainArray.length(); i++){
            JSONObject trainOb=(JSONObject)trainArray.get(i);
            String index= (String) trainOb.get("@index"); //index is the train NO.
            //System.out.println(trainOb.get("@index"));
            JSONArray stopArray=(JSONArray)trainOb.get("stop");
            for (int j=0; j<stopArray.length(); j++){
                JSONObject stopOb=(JSONObject) stopArray.get(j);
                String station= (String) stopOb.get("@station");
                String origTime= (String) stopOb.get("@origTime");
                //create a TrainSchedule object for each (trainNo + station)
                TrainSchedule schedule=new TrainSchedule( index, date, station, origTime);
                String key=index+station; //use combined string as key
                scheduleMap.put(key, schedule); //add each schedule to map
            }
        }
```

Task 2 View Code from jsp file

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%--<%= request.getAttribute("doctype") %>--%>
<%@ page import="java.util.ArrayList"%>
<%@ page import="edu.cmu.kunt.bart2.Log" %>

<%--
<!DOCTYPE html>--%>
<html>
<body>

<h1>Dashboard for BART web service</h1>
<%--because this results page is accessed by URL so only "get" method is used.--%>
<form action="dashboard" method="get">
    <h2>
        1. Total of null inputs from user is <%=request.getAttribute("nullInputs")%>.<br>
```

```
        2.  <%=request.getAttribute("api200")%> out of total
        <%=request.getAttribute("totalLogs")%> logs returned data from 3P API. <br>
        3.  <%=request.getAttribute("api200Pct")%> % of total have 200 status from 3p Api.<br>
        4. The most popular train is No.<%=request.getAttribute("trainNo")%>,
        with <%=request.getAttribute("maxTrainCount")%> times searched.

    </h2>

    <h2>All logs saved in MongoDB</h2>
    <style>
        table, th, td {
            border:1px solid black;
            border-collapse: collapse;
        }
    </style>

    <table style="width:100%">
        <tr>
            <th>Timestamp</th>
            <th>Date</th>
            <th>TrainNo</th>
            <th>Station</th>
            <th>User Input</th>
            <th>API status</th>
            <th>Reply to User</th>
        </tr>
        <% ArrayList list = (ArrayList) request.getAttribute("logs"); %>

        <% for(int i=0;i<list.size();i++){%>
        <tr>
            <%Log log= (Log) list.get(i);%>
            <td><%=log.timestamp%></td>
            <td><%=log.date%></td>
            <td><%=log.trainNo%></td>
            <td><%=log.station%></td>
            <td><%=log.input%></td>
            <td><%=log.from3pApi%></td>
            <td><%=log.toAndroid%></td>

           <%-- <td> <%= list.get(i).toString()%></td>--%>
        </tr>
```

```
        <% } %>

    </table>

</form>
</body>
</html>
```

Class Log

```
package edu.cmu.kunt.bart2;

import org.bson.types.ObjectId;
import java.sql.Timestamp;

public class Log {

    public Timestamp timestamp;
    public String date;
    public String trainNo;
    public String station;
    public String input;
    public String from3pApi;
    public String toAndroid;

    Log(
        Timestamp timestamp,
        String date,
        String trainNo,
        String station,
        String input,
        String from3pApi,
        String toAndroid){

        this.timestamp=timestamp;
        this.date=date;
        this.trainNo=trainNo;
        this.station=station;
        this.input =input;
```

```
        this.from3pApi=from3pApi;
        this.toAndroid=toAndroid;

    }

    @Override
    public String toString() {
        return
                "Timestamp= " + timestamp +
                ", Date=" + date  +
                ", TrainNo=" + trainNo  +
                ", Station=" + station  +
                ", Input=" + input  +
                ", Message from Api=" + from3pApi  +
                ", Message to Android=" + toAndroid + '\n';
    }
}
```