Project 4 Task1 Writeup- BART Web Service

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

Task 1

URL of web server from Heroku:

https://serene-stream-01792.herokuapp.com/

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

1. Implement a native Android application

a. Has at least three different kinds of Views in your Layout (TextView, EditText, ImageView, or anything that extends android.view.View). **In order to figure out if something is a View, find its API. It it extends android.view.View then it is a View.**

My android app has 3 views: TextView, EditText and RadioButton group. Details are in 'content_main.xml' and 'BartController' of project 'AndroidBART'.

b. Requires input from the user

Android requires user to input Train No and choose a Station.

c. Makes an HTTP request (using an appropriate HTTP method) to your web service

HTTP request and GET method is used in getSchdule method of ‘BARTModel’ (line 69-72) for both task 1 and 2.

d. Receives and parses an XML or JSON formatted reply from your web service

My android app receives Json reply from webservice and parse it to String type in Gson. Please see line 71 in Android project “BartController”.

e. Displays new information to the user

It displays new information about train schedule to user in responseView. Please see scheduleReady method in Android controller.

f. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.)

My App can be used repeatable without restarting it.

2. Implement a web service, deployed to Heroku

a. Implement a simple (can be a single path) API.

I use 3p API from BART government web. URL used is "https://api.bart.gov/api/sched.aspx?cmd=routesched&route=12&key=MW9S-E7SL-26DU-VV8V&json=y"

b. Receives an HTTP request from the native Android application

HTTP request is received and replied in doGET method is used in ‘BARTServlet’ for both task 1 and 2.

c. Executes business logic appropriate to your application. This includes fetching XML or JSON information from some 3rd party API and processing the response.

Business logic: fetching Json format train schedule from BART API, then parse it in JSON and GSON library for different Trains and Stations, and send the right schedule to user.

- -10 if you use a banned API
- -10 if screen scrape instead of fetching XML or JSON via a published API

d. Replies to the Android application with an XML or JSON formatted response. The schema of the response can be of your own design.

- -5 if information beyond what is needed is passed on to the Android app, forcing the mobile app to do more computing than is necessary.

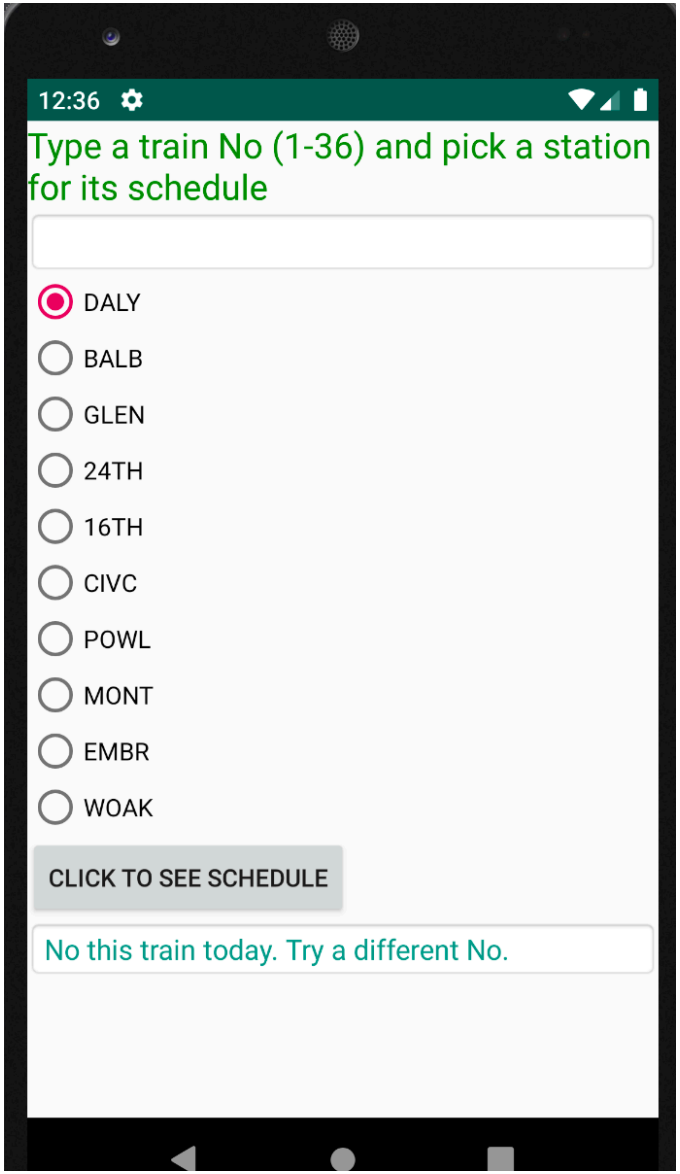
Reply is in JSON format in line 68 of BARTServlet. The reply message was good enough to present to user without any computing.

Refer back to Lab 3 for instructions on pushing a web service to Heroku.

3. Handle error conditions

Your application should test for and handle gracefully:

- Invalid mobile app input
  If user inputs null or invalid train No. as below, the app would return a message 'No this train available today. Please try a different No."


12:36
Type a train No (1-36) and pick a station for its schedule
DALY
BALB
GLEN
24TH
16TH
CIVC
POWL
MONT
EMBR
WOAK
CLICK TO SEE SCHEDULE
No this train today. Try a different No.

- Invalid server-side input (regardless of mobile app input validation):
  If server returns null TrainSchedule, it would provide a message to user.

  BARTModel code related:

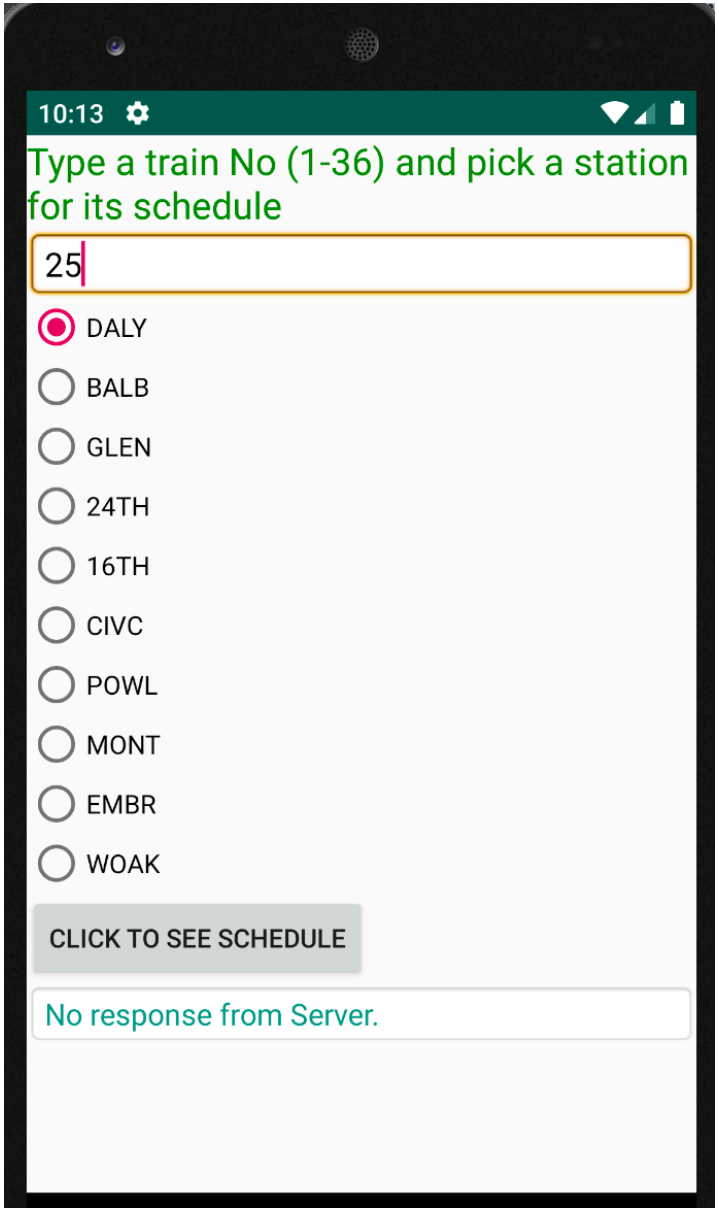
```
public static TrainSchedule getSchedule(String input){
    if (fetchData().getResponseCode()!=200){
        return null;  //4. taking care of 3p API unavailable
    }
    String fetchedData= fetchData().getResponseText();

    loadScheduleMap(fetchedData);
    if (scheduleMap.containsKey(input)){
        return scheduleMap.get(input);
    }
    else{ //5. taking care of invalid data from API or invalid input from user
        return null;
    }
}
```

  BARTServlet code related

```
returnedSchedule=BARTModel.getSchedule(input);
if (returnedSchedule==null){
    reply="No this train today. Try a different No.";
}
```

- Mobile app network failure, unable to reach server

  If server is unavailable or unconnected, the app would return a message to user without crash:


10:13
Type a train No (1-36) and pick a station for its schedule
25
DALY
BALB
GLEN
24TH
16TH
CIVC
POWL
MONT
EMBR
WOAK
CLICK TO SEE SCHEDULE
No response from Server.

- Third-party API unavailable

  This is taken care by catch exceptions and the status code from 3p API.

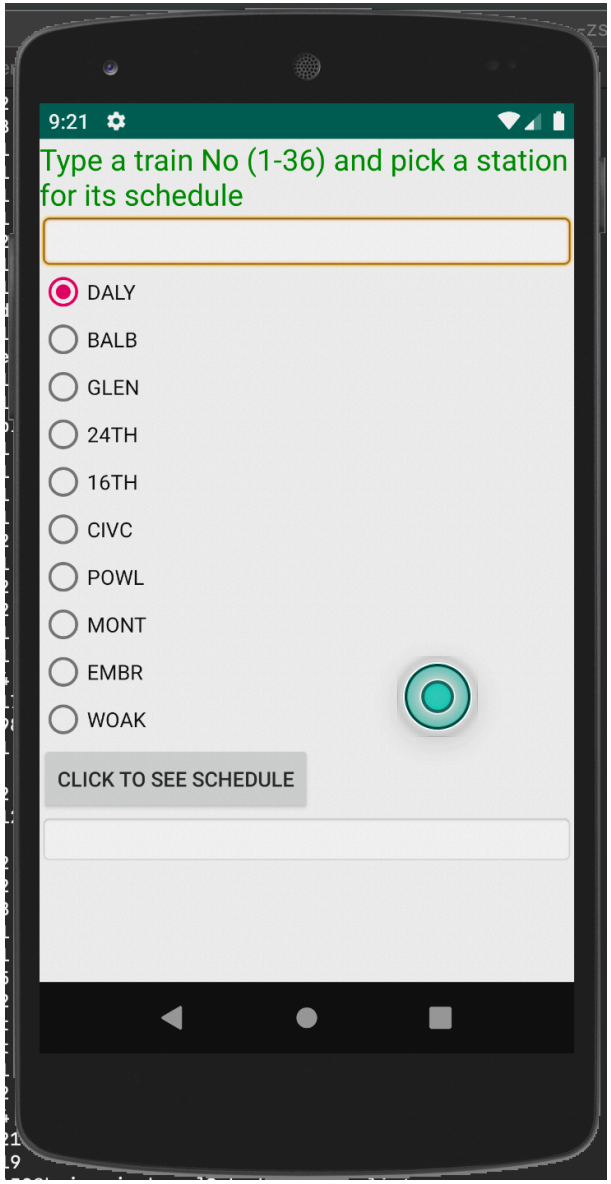
```
// Make an HTTP GET request
public static Result fetchData() {

    HttpURLConnection conn;
    int status = 0;
    Result result = new Result();
    try {

        URL url = new URL (urlString);
        //URL url=new URL("https://api.bart.gov/api/bsa.aspx?cmd=count&key=MW9S-E7SL-26DU-VV8V");
        //key: ZA2V-5LSA-9P2T-DWEI, https://api.bart.gov/api/route.aspx?cmd=routeinfo&route=8&key=MW9S-
E7SL-26DU-VV8V
        conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        // we are sending plain text
        conn.setRequestProperty("Content-Type", "text/plain; charset=utf-8");
        // tell the server what format we want back
        conn.setRequestProperty("Accept", "text/plain");

        // wait for response
        status = conn.getResponseCode();

        // set http response code
        result.setResponseCode(status);
        // set http response message - this is just a status message
        // and not the body returned by GET
        result.setResponseText(conn.getResponseMessage());

        if (status == 200) {
            String responseBody = getResponseBody(conn);
            result.setResponseText(responseBody);
        }  //4. taking care of 3p API unavailable
        else result.setResponseText("Abnormal 3p API");
    }
    // 2. handle exceptions, taking care of invalid server-side input
    catch (MalformedURLException e) {
```

```
            System.out.println("URL Exception thrown" + e);
        } catch (IOException e) {
            System.out.println("IO Exception thrown" + e);
        } catch (Exception e) {
            System.out.println("IO Exception thrown" + e);
        }
        return result;
}
```

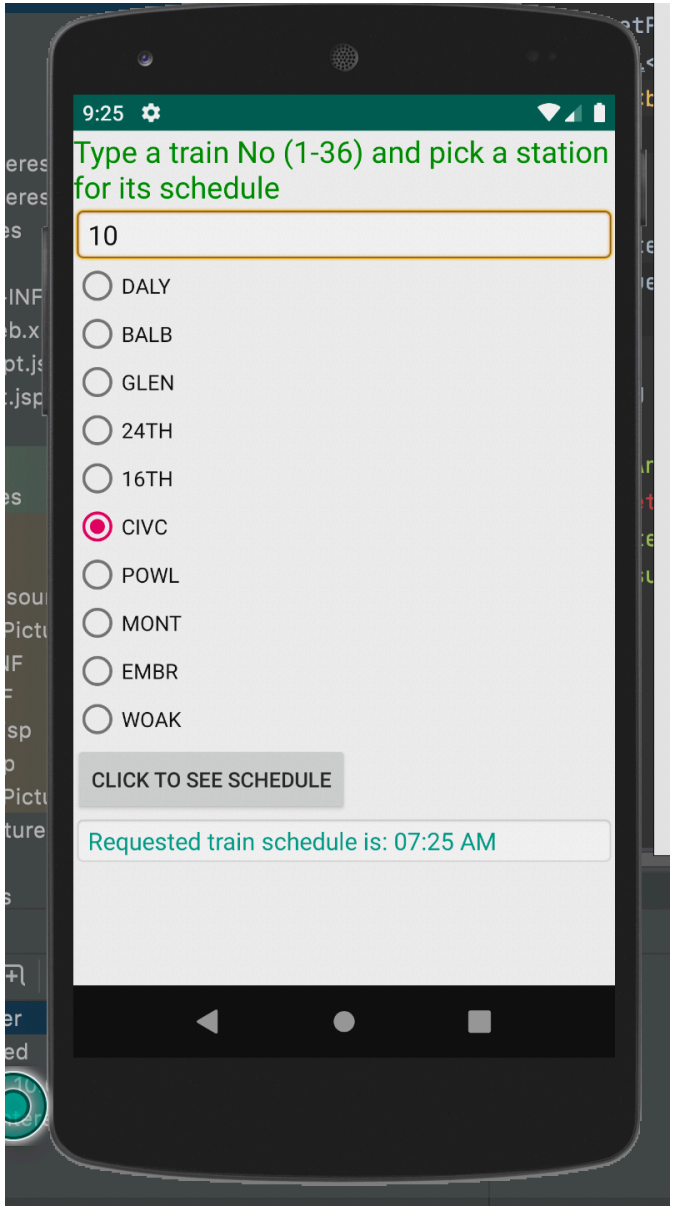
- Third-party API invalid data

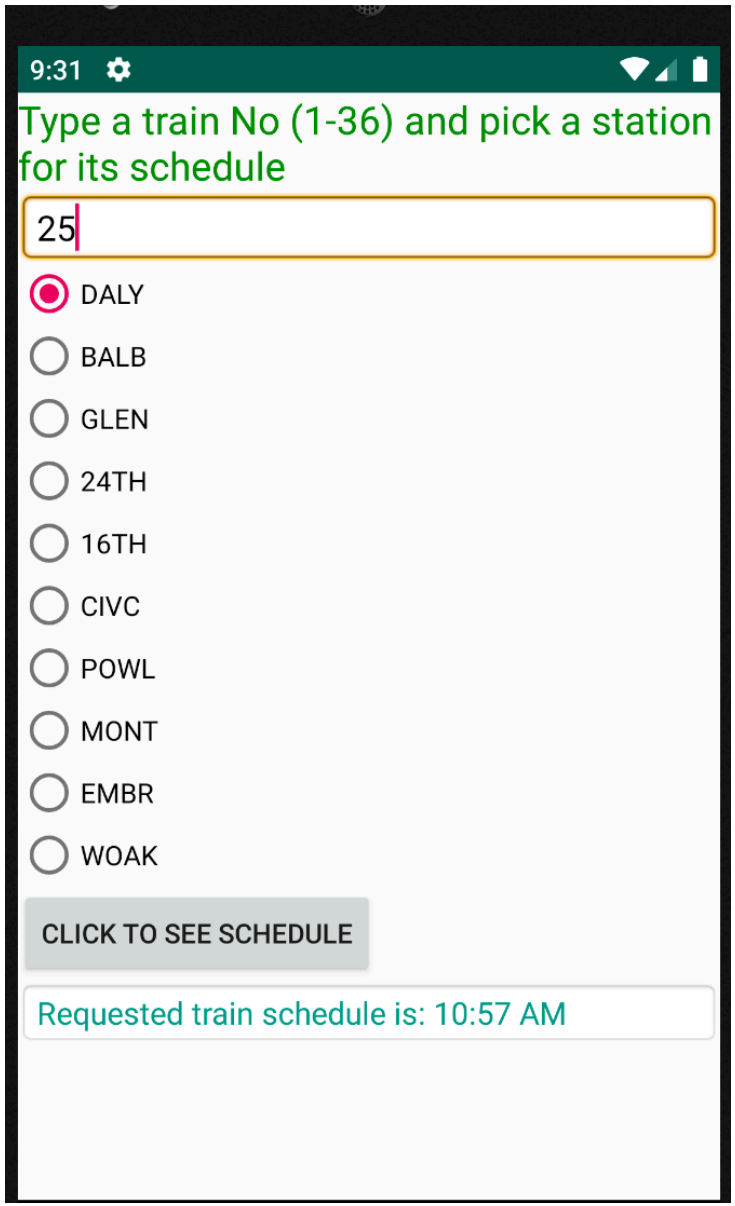
Please see the fetchData method code above, invalid data is taken care of.

The start of Android screenshot

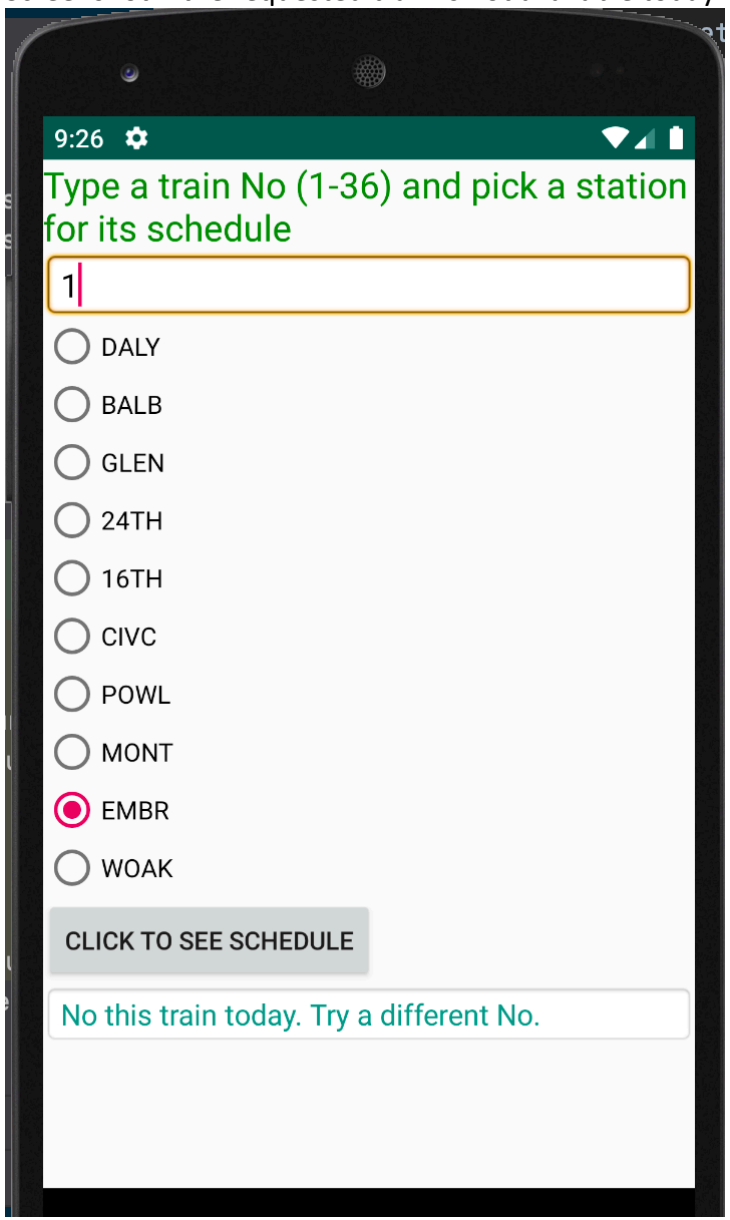

9:21
Type a train No (1-36) and pick a station for its schedule
DALY
BALB
GLEN
24TH
16TH
CIVC
POWL
MONT
EMBR
WOAK
CLICK TO SEE SCHEDULE

Screenshot of a normal search and result screen:




9:31
Type a train No (1-36) and pick a station for its schedule
25
DALY
BALB
GLEN
24TH
16TH
CIVC
POWL
MONT
EMBR
WOAK
CLICK TO SEE SCHEDULE
Requested train schedule is: 10:57 AM

Screenshot if the requested train is not available today



Task1 BART Model code

```
package ds.project4task1;

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.HashMap;
import java.util.Map;

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";

    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("\nroot: "+root);
        System.out.println("date: "+date);

        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
            }
        }
        //return scheduleMap;
    }
    // return a TrainSchedule as given input
    public static TrainSchedule getSchedule(String input){
        if (fetchData().getResponseCode()!=200){
            return null;  //4. taking care of 3p API unavailable
        }
        //call fetchData() to retrieve data from 3p Api
      String fetchedData= fetchData().getResponseText();

      loadScheduleMap(fetchedData);
      if (scheduleMap.containsKey(input)){
          return scheduleMap.get(input);
      }
      else{ //5. taking care of invalid data from API or invalid input from user
          return null;
      }
    }


    // Make an HTTP GET request  to fetch data from 3p Api
    //Citation: some syntax is from lab file provided by professor
    public static Result fetchData() {
```

```
        HttpURLConnection conn;
        int status = 0;
        Result result = new Result();
        try {

            URL url = new URL (urlString);
            //URL url=new URL("https://api.bart.gov/api/bsa.aspx?cmd=count&key=MW9S-E7SL-26DU-VV8V");
            //key: ZA2V-5LSA-9P2T-DWEI, https://api.bart.gov/api/route.aspx?cmd=routeinfo&route=8&key=MW9S-
E7SL-26DU-VV8V
            conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            // we are sending plain text
            conn.setRequestProperty("Content-Type", "text/plain; charset=utf-8");
            // tell the server what format we want back
            conn.setRequestProperty("Accept", "text/plain");

            // wait for response
            status = conn.getResponseCode();

            // set http response code
            result.setResponseCode(status);
            // set http response message - this is just a status message
            // and not the body returned by GET
            result.setResponseText(conn.getResponseMessage());

            if (status == 200) {
                String responseBody = getResponseBody(conn);
                result.setResponseText(responseBody);
            }  //4. taking care of 3p API unavailable
            else result.setResponseText("Abnormal 3p API");
        }
        // 2. handle exceptions, taking care of invalid server-side input
        catch (MalformedURLException e) {
            System.out.println("URL Exception thrown" + e);
        } catch (IOException e) {
            System.out.println("IO Exception thrown" + e);
        } catch (Exception e) {
            System.out.println("IO Exception thrown" + e);
        }
        return result;
    }
```

```
    //Citation: this is from lab file provided by professor
    // Gather up a response body from the connection
    // and close the connection.
    public static String getResponseBody(HttpURLConnection conn) {
        String responseText = "";
        try {
            String output = "";
            BufferedReader br = new BufferedReader(new InputStreamReader(
                    (conn.getInputStream())));
            while ((output = br.readLine()) != null) {
                responseText += output;
            }
            conn.disconnect();
        } catch (IOException e) {
            System.out.println("Exception caught " + e);
        }
        return responseText;
    }
}
```

Task1 BART Servlet

```
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;

@WebServlet(name = "BARTServlet",
        urlPatterns = {"/trainSchedule/*"})
public class BARTServlet extends HttpServlet {
    //public static Map<String, TrainSchedule> scheduleMap=new HashMap<>();

    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
     // Make an HTTP GET request
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("here");
        String input=null;
        Gson gson=new Gson();
        String reply = null;
        String jsonReply = "";
        PrintWriter out=response.getWriter();
        //String url = request.getServletPath();
        String url = request.getRequestURI();
        System.out.println("Url:"+url);
        //String[] inputs=url.split("trainSchedule/");
        if (url.contains("trainSchedule/")){

            String[] inputs=url.split("Schedule/");
            if (inputs.length>1) {
                input=inputs[1];
                System.out.println(input);
            }
        }
        //1.ctaking care of invalid mobile app input
        if (input==null || input.isEmpty()) {
            reply="Null input. Please submit the search term.";

            System.out.println("null input");
        }
        if (input!=null && !input.isEmpty() ){//get the schedule if available
            TrainSchedule returnedSchedule=BARTModel.getSchedule(input);
            if (returnedSchedule==null){
                reply="No this train today. Try a different No.";
            }
            else {//this is reply message to user with retrieved schedule time
                reply="Requested train schedule is: "+returnedSchedule.origTime;
            }
        }
```

```
        jsonReply=gson.toJson(reply);
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        out.print(jsonReply);
        out.close();

    }
}
```

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();

    }

}
```