Appendix:

Email With Client

Client: Battle of the Band director (Ms. Lopez)

Questions:

1. What are some BoB problems that you faced in the previous years?
2. Are these problems solvable by possibly creating an app?
3. How do you feel about an app being created to resolve these problems?

What are some BoB problems that you faced in the previous years?

A major problem area for us has always been managing our raffling system. Because we rely on a manual input system through Google Forms, sometimes we experience errors with the inputted information (names, raffle nos, etc) as we select the winners.
Also, because of our manual spinning system, some tickets get stuck in the edges of our machine and are thus less likely to be chosen, which doesn't provide a fair chance of winning to all our buyers.

The current raffle system that the BOB committee uses to select its prize winners is quite time-consuming. Additionally, not all raffle-ticket holders are given a fair chance to win because number stubs often get caught in raffle spinner's cavities (I'm not sure if this is the right word for it).

Another issue is with our Audience Choice Award voting. As audience members typically text the names of their favorite bands & vocalists to a cell number we hold, it's difficult to keep track of them as we often have to manually count the number of votes we receive.

Second, the current system in place for determining the winners of the audience choice awards is by manually counting votes sent in via text messages. Considering the vast number of attendees who vote (over 1000), the process is very tedious, has lots of room for human error, and is not the most efficient way of getting the task done.

Finally, often our event tends to run more quickly than planned this is because the bands take less time than the time we give them - causing issues with our running order of performers as we need them to be ready backstage at an earlier time than expected than what we informed them. It can difficult to contact the whole group of performers manually by phone or in person.

Fourth, we want all attendees to remain informed and up-to-date with the event. As the event tends to run ahead of schedule, a big help would be to have a program that would be capable of

sending live updates to those attending so that they will be on time to catch their favorite band's performance.

Are these problems solvable by possibly creating an app?

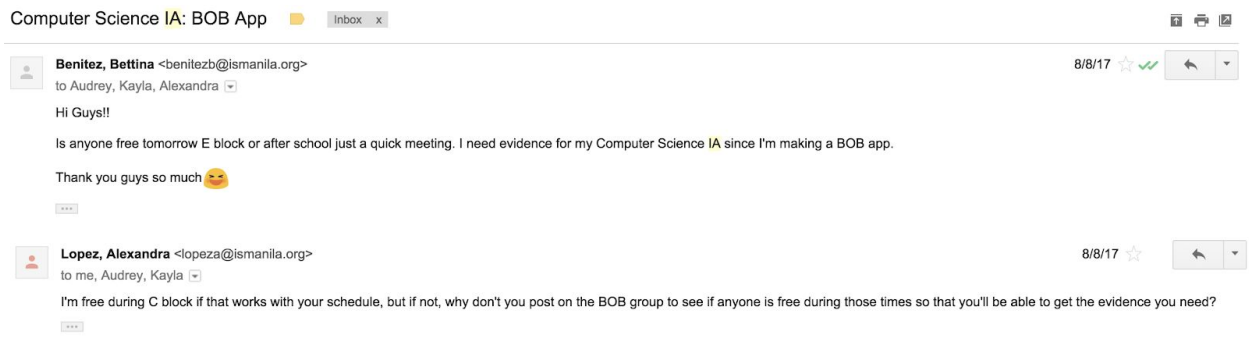
I think several of these issues could be resolved, creating a better experience for both the audience and the organizers.

On the audience's end, they'd be able to keep updated on the running order of our performers, and easily submit votes for their Audience Choice Awards.
For the BOB organizers, it'd be much more convenient to manage these above responses digitally. I think it would also be a potential solution to our raffle system if we were able to digitize these transactions, reducing human error and creating a more fair raffling system.

How do you feel about an app being created to resolve these problems?

I think it would be beneficial to our organizing committee to look into these solutions digitally through an app, and see which features would be able to streamline our operations most efficiently.

An app would help the BOB committee to a more systematized and organized way of running the event. In addition, an app for BOB would allow the committee to complete tasks more efficiently, as well as keep the event's viewers up-to-date and well-informed.



Summary of Transcript of Interview with Ms. Lopez

Me: Good Afternoon! Thanks for meeting with me!

Ms. Lopez: Good Afternoon! No, it's my pleasure!

Me: So... I'm here to talk to you about Battle of the Bands. Since you're the director of Battle of the Bands, I want to know the behind the scenes. For such a big event, you must have some issues along the way.

Ms. Lopez: Oh definitely!! There are just too many to name but the biggest problems I believe are with the schedule, the raffles, and the voting. You see, us organisers, we are running around Battle of the Bands, constantly checking everything. For finals, this is still a secret so keep it on the down low, we have a whole new layout and it's going to be much bigger than other years. While bigger is better, we know it will be harder for our organisers to be constantly updated. Our schedule is always a mess, and everyone in the BoB committee can't expect a proper schedule to run. Also, we have raffles by the raffle station. I end up having to stay there most of the night because I'm running the raffles for the crew — I end up missing my other important duties. And for the voting, the organisers are constantly running around the venue ensuring Battle of the Bands is running smoothly, that means, they don't always get to vote for their favourite performers.

Me: Wow that must be tough!

Ms. Lopez: Absolutely! It takes months of work!

Me: So as you read from my email, I'm planning to propose a product that will possibly help your event. What do you think?

Ms. Lopez: Of course! I love this idea! I think this would be a great addition to BoB, if not this year, in future years.

Me: Definitely! What type of product do you think would be most suitable for your organisers?

Ms. Lopez: Definitely something easily accessible. Like I said, BoB committee is always running around doing errands and all they have is their phone.

Me: How about a mobile app?

Ms. Lopez: That's actually perfect!! We are all required to have our phones so that is perfect for us!

...

Me: Thank you so much for meeting with me. I'll send you another message to follow up on the app!

Ms. Lopez: No, thank you! I can't wait!

——

Me: Hello again! Thanks for meeting with me!

Ms. Lopez: Oh no, it's my pleasure!

Me: So, part of starting the software develop process is creating a success criterion. This is essentially what I'd expect the app to do, and I hope you find these appropriate.
Ms. Lopez: Oh that's perfect!

...

Ms. Lopez: I actually think the app would be better first for just us organisers. That way we could all work on different functions of the app. I also would like the audience to pay more attention to performers

Me: Yes that can be fixed!

Ms. Lopez: Other than that, I think everything is perfect!! Thank you so much for doing this!

Final Consultation with Client:

My final consultation with my client occurred

Client typed comments directly onto document. All data is critique directly from client.

CRITERION:

1.
    a. The app launched the login page immediately.
    b. The aesthetic of the login page is great, very user-friendly
    c. When given the wrong email and password, an incorrect email/password sign appeared
2.
    a. The signup page is easy to use, and similar aesthetics as the login page
    b. I am able to give an admin key
    c. Password is hidden as I type
3.

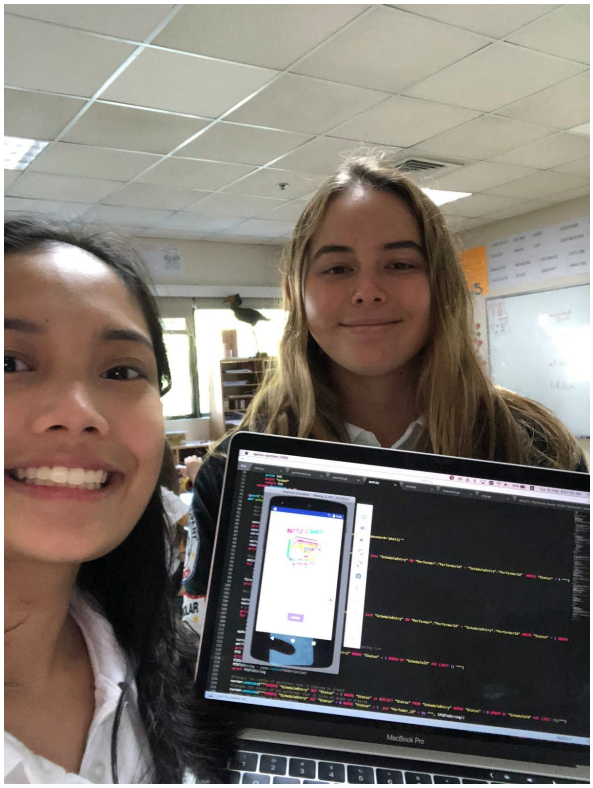
    a. The icons at the bottom of the page is easy to use, and I can easily flip back between windows

Top and bottom bars are designed with the latest Bob colours and logo – a little too much going on though

4.
    a. The schedule updater is very efficient and the schedule is correctly inputted into the database so the lineup of performers is accurate
5.
    a. App chooses a random user as a winner.
    b. The randomiser chooses from a student but it doesn't delete the already used raffle tickets
6.
    a. Users have stored raffle numbers so it's easy to retrieve first and last name.
7.
    a. Users can choose easily from a dropdown menu of bands and vocals.
    b. The app tells me whether i've placed a vote already or not.

1. Fix layout
    a. It would be great to have photos of the performers so for bands, it would be easier to know who is part of the band
    b. Maybe a list of band members would be useful too
    c. Some of the alignment should be fixed
2. Add map of the venue
    a. I realised that our new venue layout made it more difficult to find the venue
3. Voting
    a. At the end of voting period, the user should be able to see who they voted for and maybe take out the drop down menus.

Proof of consultation:



Classes:

```
129 ));
130
131                     conn.disconnect();
132                 } catch (Exception e) {
133                     e.printStackTrace();
134                 }
135             }
136         });
137
138         thread.start();
139     }
140
141
142 }
143
144
145
```

```
 93
 94      public void sendPost() {
 95          Thread thread = new Thread(new Runnable() {
 96              @Override
 97              public void run() {
 98                  try {
 99                      GmailAddress = email.getText().
    toString();
100                      FirstName = firstName.getText().
    toString();
101                      LastName = lastName.getText().toString
    ();
102                      Password = password.getText().toString
    ();
103                      UserType = userType.getText().toString
    ();
104
105
106
107                      URL url = new URL("http://10.0.2.2:
    8080/postdata");
108                      HttpURLConnection conn = (
    HttpURLConnection) url.openConnection();
109                      conn.setRequestMethod("POST");
110                      conn.setRequestProperty("Content-Type"
    , "application/json;charset=UTF-8");
111                      conn.setRequestProperty("Accept","
    application/json");
112                      conn.setDoOutput(true);
113                      conn.setDoInput(true);
114
115                      JSONObject jsonParam = new JSONObject(
    );
116                      jsonParam.put("GmailAddress",
    GmailAddress);
117                      jsonParam.put("FirstName", FirstName);
118                      jsonParam.put("LastName", LastName);
119                      jsonParam.put("Password", Password);
120                      jsonParam.put("UserType", UserType);
121
122                      DataOutputStream os = new
    DataOutputStream(conn.getOutputStream());
123                      os.writeBytes(jsonParam.toString());
124
125                      os.flush();
126                      os.close();
127
128                      Log.i("STATUS", String.valueOf(conn.
    getResponseCode()));
129                      Log.i("MSG" , conn.getResponseMessage(
```

```
50          email = (EditText) findViewById(R.id.emails);
51          firstName = (EditText) findViewById(R.id.firstname)
    ;
52          lastName = (EditText) findViewById(R.id.lastname);
53          password = (EditText) findViewById(R.id.password);
54          userType = (EditText) findViewById(R.id.usertype);
55          admin = (TextView) findViewById(R.id.textView10);
56
57          button4 = (Button) findViewById(R.id.button4);
58          button4.setOnClickListener(new View.OnClickListener
    () {
59              @Override
60              public void onClick(View view) {
61                  sendPost();
62                  if (isConnectedToServer() == false) {
63                      admin.setText("ERROR");
64                  }
65                  email.setText("");
66                  firstName.setText("");
67                  lastName.setText("");
68                  password.setText("");
69                  userType.setText("");
70                  Intent intent = new Intent(SignUpActivity.
    this, LoginActivity.class);
71                  startActivity(intent);
72
73              }
74          });
75
76      }
77
78      public boolean isConnectedToServer() {
79          try{
80              URL url = new URL("http://" + ipAddress + "/
    postdata");
81              HttpURLConnection conn = (HttpURLConnection)
    url.openConnection();
82              conn.setRequestMethod("POST");
83              conn.setRequestProperty("Content-Type", "
    application/json;charset=UTF-8");
84              conn.setRequestProperty("Accept","application/
    json");
85              conn.setDoOutput(true);
86              conn.setDoInput(true);
87              return true;
88          } catch (Exception e) {
89              return false;
90          }
91      }
92
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/SignUpActivity

```java
package com.example.bettinabenitez.bobapp;

import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.content.Intent;
import android.util.Log;

import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
import java.net.URL;
import java.net.HttpURLConnection;
import java.io.Writer;
import java.io.DataOutputStream;
import java.io.BufferedWriter;
import java.io.OutputStreamWriter;
import java.io.FileWriter;

import org.json.JSONObject;
import org.json.JSONException;

import static junit.framework.Assert.assertEquals;


public class SignUpActivity extends AppCompatActivity {

    EditText email, firstName, lastName, password, userType
;
    String FirstName, GmailAddress;
    String LastName;
    String Password, Email, UserType;
    TextView admin;
    String ipAddress = "10.70.0.180:8080" ;

    Button button4;

    //
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_signup);
```

```
121                         access = 1; //moves to the next
    activity
122                     }
123
124                     bufferedReader.close();
125
126                     Log.i("STATUS", String.valueOf(conn.
    getResponseCode()));
127                     Log.i("MSG", conn.getResponseMessage()
    );
128
129                     conn.disconnect();
130                 } catch (Exception e) {
131                     e.printStackTrace();
132                 }
133             }
134         });
135
136         thread.start();
137     }
138 }
139
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/LoginActivity.j

```
 87                      //Converting Strings into JSON format
    and adding them to the jsonParam Obeject
 88                      JSONObject jsonParam = new JSONObject(
    );
 89                      jsonParam.put("GmailAddress",
    GmailAddress);
 90                      jsonParam.put("Password", Password);
 91
 92                      //Sending JSON Objects to HTTP
    SERIALISATION
 93                      DataOutputStream outputStream = new
    DataOutputStream(conn.getOutputStream());
 94                      outputStream.writeBytes(jsonParam.
    toString());
 95
 96                      outputStream.flush();
 97                      outputStream.close();
 98
 99                      //Retrieves JSON Objects from HTTP
    DESERIALISATION
100                      InputStream inputStream = conn.
    getInputStream();
101                      BufferedReader bufferedReader = new
    BufferedReader(new InputStreamReader(inputStream));
102
103                      //Creates a string builder used to
    convert JSON data to string
104                      StringBuilder sb = new StringBuilder()
    ;
105                      String line;
106                      //Reads the JSON data
107                      line = bufferedReader.readLine();
108
109                      //checks if no JSON data was send,
    access is denied
110                      if (line == null) {
111                          access = 0; // cannot move on to
    the next activity
112                          Log.e("ACCESS", "denied");
113                          signup.setText("Incorrect Email
    and/or Password");
114                      }
115
116                      //while line is not = null, add the
    JSON data to the string builder
117                      while (line != null) {
118                          sb.append(line+"\n");
119                          Log.e("ACCESS:", line); //checks
    the access through logcat
120                          line = bufferedReader.readLine();
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/LoginActivity.j

```
48                  startActivity(intent);
49
50              }
51          });
52
53          loginbtn.setOnClickListener(new View.
    OnClickListener() {
54              @Override
55              public void onClick(View view) { //when button
    is clicked
56                  user.setEmailAddress(email.getText().
    toString());
57                  getPost(); //send and get json data
58                  if (access == 1) {
59                      Intent intent = new Intent(
    LoginActivity.this, MainActivity.class);
60                      startActivity(intent);
61                  }
62
63              }
64          });
65      }
66
67      public void getPost() {
68
69          Thread thread = new Thread(new Runnable() {
70              @Override
71              public void run() {
72                  try {
73                      GmailAddress = email.getText().toString
    ();
74                      Password = password.getText().toString(
    );
75
76
77                      //CONNECTING TO URL --> send/retrieve
    JSON data
78                      URL url = new URL("http://10.0.2.2:8080
    /login"); //10.0.2.2 => localhost
79                      HttpURLConnection conn = (
    HttpURLConnection) url.openConnection();
80                      //POST request method, from python code
81                      conn.setRequestMethod("POST");
82                      conn.setRequestProperty("Content-Type",
     "application/json;charset=UTF-8");
83                      conn.setRequestProperty("Accept", "
    application/json");
84                      conn.setDoOutput(true);
85                      conn.setDoInput(true);
86
```

```java
package com.example.bettinabenitez.bobapp;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.content.Intent;

import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.HttpURLConnection;

public class LoginActivity extends AppCompatActivity {

    Button loginbtn;
    public static TextView signup;
    EditText email, password;

    String GmailAddress, Password;
    User user = new User();
    int access = 0;

    private static String url = "http://10.0.2.2:8080/login
";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);


        signup = (TextView) findViewById(R.id.signup);
        loginbtn = (Button) findViewById(R.id.login);
        email = (EditText) findViewById(R.id.email);
        password = (EditText) findViewById(R.id.password);

        signup.setOnClickListener(new View.OnClickListener(
) {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(LoginActivity.
this, SignUpActivity.class);
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/MainActivity.j

```
78      @Override
79      public boolean onNavigationItemSelected(@NonNull
    MenuItem item) {
80          return false;
81      }
82
83
84  }
85
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/MainActivity.j

```
40                  case R.id.voting:
41                      Voting fragment3 = new Voting();
42                      FragmentTransaction
   fragmentTransaction3 = getSupportFragmentManager().
   beginTransaction();
43                      fragmentTransaction3.replace(R.id.home,
    fragment3, "FragmentName");
44                      fragmentTransaction3.commit();
45                      return true;
46                  case R.id.live_stream:
47                      LiveStream fragment4 = new LiveStream()
   ;
48                      FragmentTransaction
   fragmentTransaction4 = getSupportFragmentManager().
   beginTransaction();
49                      fragmentTransaction4.replace(R.id.home,
    fragment4, "FragmentName");
50                      fragmentTransaction4.commit();
51                      return true;
52              }
53              return false;
54          }
55
56      };
57
58      Button click;
59      public static TextView data;
60
61
62      @Override
63      protected void onCreate(Bundle savedInstanceState) {
64          super.onCreate(savedInstanceState);
65          setContentView(R.layout.activity_main);
66
67          BottomNavigationView navigation = (
   BottomNavigationView) findViewById(R.id.bottom_nv);
68          navigation.setOnNavigationItemSelectedListener(
   mOnNavigationItemSelectedListener);
69          Schedule fragment = new Schedule();
70          android.support.v4.app.FragmentTransaction
   fragmentTransaction = getSupportFragmentManager().
   beginTransaction();
71          fragmentTransaction.replace(R.id.home, fragment, "
   FragmentName");
72          fragmentTransaction.commit();
73
74
75
76      }
77
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/MainActivity.j

```
 1 package com.example.bettinabenitez.bobapp;
 2
 3 import android.support.v4.app.FragmentTransaction;
 4 import android.support.design.widget.NavigationView;
 5 import android.support.design.widget.BottomNavigationView;
 6 import android.view.MenuItem;
 7 import android.support.annotation.NonNull;
 8
 9
10 import android.support.v7.app.AppCompatActivity;
11 import android.os.Bundle;
12 import android.widget.TextView;
13 import android.widget.Button;
14
15
16 public class MainActivity extends AppCompatActivity
   implements NavigationView.OnNavigationItemSelectedListener
   {
17
18     Button update;
19     int access = 1;
20     public String url;
21     TextView performerNext, performerNow;
22
23     private BottomNavigationView.
   OnNavigationItemSelectedListener
   mOnNavigationItemSelectedListener = new
   BottomNavigationView.OnNavigationItemSelectedListener() {
24
25         @Override
26         public boolean onNavigationItemSelected(@NonNull
   MenuItem item) {
27             switch (item.getItemId()) {
28                 case R.id.schedule:
29                     Schedule fragment = new Schedule();
30                     FragmentTransaction fragmentTransaction
    = getSupportFragmentManager().beginTransaction();
31                     fragmentTransaction.replace(R.id.home,
   fragment, "FragmentName");
32                     fragmentTransaction.commit();
33                     return true;
34                 case R.id.raffle:
35                     Raffle fragment2 = new Raffle();
36                     FragmentTransaction
   fragmentTransaction2 = getSupportFragmentManager().
   beginTransaction();
37                     fragmentTransaction2.replace(R.id.home,
    fragment2, "FragmentName");
38                     fragmentTransaction2.commit();
39                     return true;
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/LiveStream.ja

```java
package com.example.bettinabenitez.bobapp;


import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;


/**
 * A simple {@Link Fragment} subclass.
 */
public class LiveStream extends Fragment {


    public LiveStream() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater,
ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.
fragment_live_stream, container, false);
    }

}

```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/Schedule.java

```
123                     break;
124             }
125
126         }
127
128 }
129
```

```
 84 JSONArray(line);
 85                                  JSONObject jsonObject =
    new JSONObject();
 86
 87                                  try {
 88                                      jsonObject.put("
    Performer_Name", jsonArray);
 89                                  }
 90                                  catch (JSONException e) {
 91                                      e.printStackTrace();
 92                                  }
 93
 94                                  JSONArray performerArray =
     jsonObject.getJSONArray("Performer_Name");
 95                                  String performingNow =
    performerArray.getString(0);
 96                                  String performingNext =
    performerArray.getString(1);
 97
 98                                  //String json = jsonObject
    .getString("Performer_Name");
 99
100                                  Log.e("ERR", performingNow
    );
101                                  performerNow.setText(
    performingNow);
102                                  performerNext.setText(
    performingNext);
103
104                                  if (line != null) {
105                                      sb.append(line+"\n");
106                                      Log.e("ERR", line);
107                                  }
108
109                                  bufferedReader.close();
110
111                                  Log.i("STATUS", String.
    valueOf(conn.getResponseCode()));
112                                  Log.i("MSG", conn.
    getResponseMessage());
113
114                                  conn.disconnect();
115                              } catch (Exception e) {
116                                  e.printStackTrace();
117                              }
118                          }
119                      });
120                      thread.start();
121                  }
122
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/Schedule.java

```
48
49          update = (Button) v.findViewById(R.id.update);
50          performerNow = (TextView) v.findViewById(R.id.
    PerformerNow);
51          performerNext = (TextView) v.findViewById(R.id.
    PerformerNext);
52          update.setOnClickListener(this);
53
54          return v;
55
56      }
57
58      @Override
59      public void onClick(View v) { //when button is clicked
60          switch (v.getId()) {
61              case R.id.update:
62                  if (access == 1) {
63                      Thread thread = new Thread(new Runnable
    () {
64                          @Override
65                          public void run() {
66                              try {
67  //                                connects to the url
68                                  URL url = new URL("http://
    10.0.2.2:8080/schedule");
69                                  HttpURLConnection conn = (
    HttpURLConnection) url.openConnection();
70  //
71                                  conn.setRequestMethod("GET"
    );
72                                  conn.setRequestProperty("
    Content-Type", "application/json;charset=UTF-8");
73                                  conn.setRequestProperty("
    Accept", "application/json");
74                                  conn.setDoOutput(true);
75                                  conn.setDoInput(true);
76
77
78                                  //Deserialisation
79                                  InputStream inputStream =
    conn.getInputStream();
80                                  BufferedReader
    bufferedReader = new BufferedReader(new InputStreamReader(
    inputStream));
81
82                                  StringBuilder sb = new
    StringBuilder();
83                                  String line =
    bufferedReader.readLine();
84                                  JSONArray jsonArray = new
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/Schedule.java

```java
package com.example.bettinabenitez.bobapp;


import android.os.Bundle;
import android.support.constraint.ConstraintLayout;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.TextView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;


/**
 * A simple {@Link Fragment} subclass.
 */
public class Schedule extends Fragment implements View.
OnClickListener {

    Button update;
    int access = 1;
    public String url;
    TextView performerNext, performerNow;
    int click = 0;

    public Schedule() {
        // Required empty public constructor
    }



    @Override
    public View onCreateView(LayoutInflater inflater,
ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View v = inflater.inflate(R.layout.
fragment_schedule, container, false);
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/Voting.java

```
127 , "application/json;charset=UTF-8");
128                         conn.setRequestProperty("Accept", "
    application/json");
129                         conn.setDoOutput(true);
130                         conn.setDoInput(true);
131
132                         //Converting Strings into JSON format
    and adding them to the jsonParam Obeject
133                         JSONObject jsonParam = new JSONObject(
    );
134                         jsonParam.put("GmailAddress",
    GmailAddress);
135                         jsonParam.put("BandVote", BandVote);
136                         jsonParam.put("VocalVote", VocalVote);
137
138                         //Sending JSON Objects to HTTP
    SERIALISATION
139                         DataOutputStream outputStream = new
    DataOutputStream(conn.getOutputStream());
140                         outputStream.writeBytes(jsonParam.
    toString());
141
142                         outputStream.flush();
143                         outputStream.close();
144
145                         Log.i("STATUS", String.valueOf(conn.
    getResponseCode()));
146                         Log.i("MSG", conn.getResponseMessage()
    );
147
148                         conn.disconnect();
149                     } catch (Exception e) {
150                         e.printStackTrace();
151                     }
152                 }
153             });
154
155             thread.start();
156         }
157 }
158
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/Voting.java

```
 87
 88      public void onItemSelected(AdapterView<?> parent, View
     view, int pos, long id) {
 89          // An item was selected. You can retrieve the
    selected item using
 90          parent.getItemAtPosition(pos);
 91          Log.e("ParentId", Integer.toString(parent.getId())
    );
 92          switch (parent.getId()){
 93              case 2131624104:
 94                  user.setVocalPerformer(parent.
    getItemAtPosition(pos).toString());
 95                  Log.e("BAND", user.getVocalPerformer());
 96                  Log.e("email", user.getEmailAddress());
 97                  break;
 98
 99              case 2131624108:
100                  user.setBandPerformer(parent.
    getItemAtPosition(pos).toString());
101                  Log.e("BAND", user.getBandPerformer());
102                  break;
103          }
104      }
105
106      public void onNothingSelected(AdapterView<?> parent) {
107          // Another interface callback
108      }
109
110      public void setVote() {
111
112          Thread thread = new Thread(new Runnable() {
113              @Override
114              public void run() {
115                  try {
116
117                      GmailAddress = user.getEmailAddress();
118                      BandVote = user.getBandPerformer();
119                      VocalVote = user.getVocalPerformer();
120
121
122                      //CONNECTING TO URL --> send/retrieve
    JSON data
123                      URL url = new URL("http://10.0.2.2:
    8080/voting"); //10.0.2.2 => localhost
124                      HttpURLConnection conn = (
    HttpURLConnection) url.openConnection();
125                      //POST request method, from python
    code
126                      conn.setRequestMethod("POST");
127                      conn.setRequestProperty("Content-Type"
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/Voting.java

```
48          vote = (Button) v.findViewById(R.id.button3);
49          message = v.findViewById(R.id.errormessage);
50
51          Spinner spinner = (Spinner) v.findViewById(R.id.
    spinnerBand);
52          spinner.setOnItemSelectedListener(this);
53          ArrayAdapter<CharSequence> adapter;
54          //R.layout.spinner_bands is the layout I created
    see PATH: res > layout > spinner_vocals
55          adapter = ArrayAdapter.createFromResource(
    getActivity().getApplicationContext(), R.array.Bands, R.
    layout.spinner_bands);
56          // Specify the layout to use when the list of
    choices appears
57          adapter.setDropDownViewResource(android.R.layout.
    simple_spinner_dropdown_item);
58          // Apply the adapter to the spinner
59          spinner.setAdapter(adapter);
60
61          Spinner spinner2 = (Spinner) v.findViewById(R.id.
    spinnerVocal);
62          spinner2.setOnItemSelectedListener(this);
63          ArrayAdapter<CharSequence> adapter2 = ArrayAdapter.
    createFromResource(getActivity().getApplicationContext(), R
    .array.Vocals, R.layout.spinner_vocals);
64          adapter2.setDropDownViewResource(android.R.layout.
    simple_spinner_dropdown_item);
65          spinner2.setAdapter(adapter2);
66
67          vote.setOnClickListener(this);
68
69          return v;
70      }
71
72      @Override
73      public void onClick(View v) {
74          if (v.getId() == R.id.button3) {
75              setVote();
76
77              if (count == 0) {
78                  message.setText("You've placed your vote");
79                  count = 1;
80              }
81              else if (count == 1) {
82                  message.setText("You've already placed your
     vote");
83              }
84
85          }
86      }
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/Voting.java

```java
package com.example.bettinabenitez.bobapp;


import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.Spinner;
import android.content.Context;
import android.widget.TextView;

import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

/**
 * A simple {@Link Fragment} subclass.
 */
public class Voting extends Fragment implements View.
OnClickListener, AdapterView.OnItemSelectedListener{

    Button vote;
    TextView message;
    String GmailAddress, BandVote, VocalVote;
    int count = 0;
    User user = new User();

    public Voting() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater,
ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View v = inflater.inflate(R.layout.fragment_voting,
 container, false);

```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/Raffle.java

```
 85
 86                                      try {
 87                                          jsonObject.put("
    RaffleWinner", jsonArray);
 88                                      }
 89                                      catch (JSONException e) {
 90                                          e.printStackTrace();
 91                                      }
 92
 93                                      JSONArray performerArray =
     jsonObject.getJSONArray("RaffleWinner");
 94                                      String gWinner =
    performerArray.getString(0);
 95                                      String regwinner =
    performerArray.getString(1);
 96
 97
 98                                      grandWinner.setText(
    gWinner);
 99                                      regularWinner.setText(
    regwinner);
100
101                                     if (line != null) {
102                                         sb.append(line+"\n");
103                                         Log.e("ERR", line);
104                                     }
105
106                                     bufferedReader.close();
107
108                                     Log.i("STATUS", String.
    valueOf(conn.getResponseCode()));
109                                     Log.i("MSG", conn.
    getResponseMessage());
110
111                                     conn.disconnect();
112                                 } catch (Exception e) {
113                                     e.printStackTrace();
114                                 }
115                             }
116                         });
117                         thread.start();
118                     }
119
120                     break;
121             }
122
123         }
124 }
125
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/Raffle.java

```
48          grandWinner = v.findViewById(R.id.grandWinner);
49          regularButton = v.findViewById(R.id.regularButton);
50          regularWinner = v.findViewById(R.id.regularWinner);
51          grandButton.setOnClickListener(this);
52          regularButton.setOnClickListener(this);
53
54          return v;
55      }
56
57      @Override
58      public void onClick(View v) {
59          switch (v.getId()) {
60              case R.id.grandButton:
61                  if (access == 1) {
62                      Thread thread = new Thread(new Runnable
    () {
63                          @Override
64                          public void run() {
65                              try {
66  //                                connects to the url
67                                  URL url = new URL("http://
    10.0.2.2:8080/raffle");
68                                  HttpURLConnection conn = (
    HttpURLConnection) url.openConnection();
69  //
70                                  conn.setRequestMethod("GET"
    );
71                                  conn.setRequestProperty("
    Content-Type", "application/json;charset=UTF-8");
72                                  conn.setRequestProperty("
    Accept", "application/json");
73                                  conn.setDoOutput(true);
74                                  conn.setDoInput(true);
75
76
77                                  //Deserialisation
78                                  InputStream inputStream =
    conn.getInputStream();
79                                  BufferedReader
    bufferedReader = new BufferedReader(new InputStreamReader(
    inputStream));
80
81                                  StringBuilder sb = new
    StringBuilder();
82                                  String line =
    bufferedReader.readLine();
83                                  JSONArray jsonArray = new
    JSONArray(line);
84                                  JSONObject jsonObject = new
     JSONObject();
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/Raffle.java

```java
package com.example.bettinabenitez.bobapp;


import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;


/**
 * A simple {@Link Fragment} subclass.
 */
public class Raffle extends Fragment implements View.
OnClickListener {

    Button grandButton, regularButton;
    int access = 1;
    int gclick = 0;
    int rclick = 0;
    public String url;
    TextView grandWinner, regularWinner;

    public Raffle() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater,
ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View v = inflater.inflate(R.layout.fragment_raffle,
 container, false);

        grandButton = v.findViewById(R.id.grandButton);
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/trial.java

```
 1 package com.example.bettinabenitez.bobapp;
 2
 3 /**
 4  * Created by bettinabenitez on 29/01/2018.
 5  */
 6
 7 import java.io.FileWriter;
 8
 9 import org.json.simple.JSONArray;
10 import org.json.simple.JSONObject;
11
12
13 public class trial {
14     public static void main(String[] args) {
15
16         JSONObject jsonObject = new JSONObject();
17
18         //JSON object and values
19         jsonObject.put("name", "Atul Rai");
20         jsonObject.put("occupation", "Blogger");
21         jsonObject.put("location", "India");
22         jsonObject.put("website", "www.websparrow.org");
23
24         //JSON array and values
25         JSONArray jsonArray = new JSONArray();
26         jsonArray.add("Java");
27         jsonArray.add("Struts");
28         jsonArray.add("jQuery");
29         jsonArray.add("JavaScript");
30         jsonArray.add("Database");
31
32         jsonObject.put("technology", jsonArray);
33
34         // writing the JSONObject into a file(info.json)
35         try {
36             FileWriter fileWriter = new FileWriter("info.
   json");
37             fileWriter.write(jsonObject.toJSONString());
38             fileWriter.flush();
39         } catch (Exception e) {
40             e.printStackTrace();
41         }
42         System.out.println(jsonObject);
43     }
44
45 }
46
```

File - /Users/bettinabenitez/Downloads/Forms 2/Product/BobApp/app/src/main/java/com/example/bettinabenitez/bobapp/User.java

```
 1 package com.example.bettinabenitez.bobapp;
 2
 3 import android.util.Log;
 4 /**
 5  * Created by bettinabenitez on 19/02/2018.
 6  */
 7
 8 public class User {
 9
10     private static String emailAddress;
11     private String bandPerformer;
12     private String vocalPerformer;
13
14
15     public static String getEmailAddress() {
16         return emailAddress;
17     }
18
19     public String getBandPerformer() {
20         return bandPerformer;
21     }
22
23     public String getVocalPerformer() { return
   vocalPerformer; }
24
25     public void setEmailAddress(String emailAddress) {
26         this.emailAddress = emailAddress;
27     }
28
29
30     public void setBandPerformer(String bandPerformer) {
31         this.bandPerformer = bandPerformer;
32     }
33
34     public void setVocalPerformer(String vocalPerformer) {
35         this.vocalPerformer = vocalPerformer;
36     }
37 }
38
```

SQL DUMP

```
toc.dat0000600 0004000 0002000 00000020543 13242700632 0014442
0ustar00postgrespostgres0000000 0000000 PGDMP
vbobapp10.110.1![00ENCODINGENCODINGSET client_encoding = 'UTF8';
false\00
STDSTRINGS
STDSTRINGS(SET standard_conforming_strings = 'on';
false]126216397bobappDATABASEdCREATE DATABASE bobapp WITH TEMPLATE = template0
ENCODING = 'UTF8' LC_COLLATE = 'C' LC_CTYPE = 'C';
DROP DATABASE bobapp;
postgresfalse26152200publicSCHEMACREATE SCHEMA public;
DROP SCHEMA public;
postgresfalse^00
SCHEMA publicCOMMENT6COMMENT ON SCHEMA public IS 'standard public schema';
postgresfalse3307913241plpgsql          EXTENSION?CREATE EXTENSION IF NOT EXISTS plpgsql WITH
SCHEMA pg_catalog;
DROP EXTENSION plpgsql;
false_00EXTENSION plpgsqlCOMMENT@COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural
language';
false1À125916514bandvote_id_seqSEQUENCEqCREATE SEQUENCE bandvote_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
&DROP SEQUENCE public.bandvote_id_seq;
publicpostgresfalse3ƒ125916409BandVoteTABLE≤CREATE TABLE "BandVote" (
    "BandVoteId" integer DEFAULT nextval('bandvote_id_seq'::regclass) NOT NULL,
    "User_Email" character(255),
    "Performer_Name" character(255)
);
DROP TABLE public."BandVote";
publicpostgresfalse2033∆125916419      PerformerTABLEäCREATE TABLE "Performer" (
    "PerformerId" integer NOT NULL,
    "Performer_Name" character(255),
    "PerformerType" character(255)
);
DROP TABLE public."Performer";
publicpostgresfalse3»125916432RaffleTicketTABLE†CREATE TABLE "RaffleTicket" (
    "RaffleId" integer NOT NULL,
    "UserId" integer,
    "Status" integer DEFAULT 0,
    "RaffleTicketNumber" character(255)
```

```
);
"DROP TABLE public."RaffleTicket";
publicpostgresfalse3«125916425
ScheduleEntryTABLEóCREATE TABLE "ScheduleEntry" (
    "ScheduleId" integer NOT NULL,
    "PerformerId" integer,
    "Status" integer,
    "Perfomer_id" character(255)
);
#DROP TABLE public."ScheduleEntry";
publicpostgresfalse3 125916494UserTABLEÓCREATE TABLE "User" (
    "UserId" integer NOT NULL,
    "GmailAddress" character(255),
    "FirstName" character(255),
    "LastName" character(255),
    "Password" character(255),
    "UserType" character(255),
    "User_Id" integer
);
DROP TABLE public."User";
publicpostgresfalse3...125916492User_UserId_seqSEQUENCEÇCREATE SEQUENCE "User_UserId_seq"
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
(DROP SEQUENCE public."User_UserId_seq";
publicpostgresfalse3202`00User_UserId_seqSEQUENCE OWNED BY;ALTER SEQUENCE
"User_UserId_seq" OWNED BY "User"."UserId";
publicpostgresfalse201Ã125916523vocalvote_id_seqSEQUENCErCREATE SEQUENCE vocalvote_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
'DROP SEQUENCE public.vocalvote_id_seq;
publicpostgresfalse3≈125916414          VocalVoteTABLEµCREATE TABLE "VocalVote" (
    "VocalVoteId" integer DEFAULT nextval('vocalvote_id_seq'::regclass) NOT NULL,
    "User_Email" character(255),
    "Performer_Name" character(255)
);
DROP TABLE public."VocalVote";
```

publicpostgresfalse2043
260416497
User UserIdDEFAULTbALTER TABLE ONLY "User" ALTER COLUMN "UserId" SET DEFAULT
nextval('"User_UserId_seq"'::regclass);
>ALTER TABLE public."User" ALTER COLUMN "UserId" DROP DEFAULT;
publicpostgresfalse201202202P016409BandVote
TABLE DATAKCOPY "BandVote" ("BandVoteId", "User_Email", "Performer_Name") FROM stdin;
publicpostgresfalse1963152.datR016419 Performer
TABLE DATAPCOPY "Performer" ("PerformerId", "Performer_Name", "PerformerType") FROM stdin;
publicpostgresfalse1983154.datT016432RaffleTicket
TABLE DATAWCOPY "RaffleTicket" ("RaffleId", "UserId", "Status", "RaffleTicketNumber") FROM stdin;
publicpostgresfalse2003156.datS016425
ScheduleEntry
TABLE DATAXCOPY "ScheduleEntry" ("ScheduleId", "PerformerId", "Status", "Perfomer_id") FROM stdin;
publicpostgresfalse1993155.datV016494User
TABLE DATAoCOPY "User" ("UserId", "GmailAddress", "FirstName", "LastName", "Password",
"UserType", "User_Id") FROM stdin;
publicpostgresfalse2023158.datQ016414 VocalVote
TABLE DATAMCOPY "VocalVote" ("VocalVoteId", "User_Email", "Performer_Name") FROM stdin;
publicpostgresfalse1973153.data00User_UserId_seqSEQUENCE SET9SELECT
pg_catalog.setval('"User_UserId_seq"', 13, true);
publicpostgresfalse201b00bandvote_id_seqSEQUENCE SET6SELECT
pg_catalog.setval('bandvote_id_seq', 7, true);
publicpostgresfalse203c00vocalvote_id_seqSEQUENCE SET7SELECT
pg_catalog.setval('vocalvote_id_seq', 5, true);
publicpostgresfalse204Ã
260616413BandVote BandVote_pkey
CONSTRAINT[ALTER TABLE ONLY "BandVote"
  ADD CONSTRAINT "BandVote_pkey" PRIMARY KEY ("BandVoteId");
DALTER TABLE ONLY public."BandVote" DROP CONSTRAINT "BandVote_pkey";
publicpostgresfalse196–
260616431Performer Performer_pkey
CONSTRAINT^ALTER TABLE ONLY "Performer"
  ADD CONSTRAINT "Performer_pkey" PRIMARY KEY ("PerformerId");
FALTER TABLE ONLY public."Performer" DROP CONSTRAINT "Performer_pkey";
publicpostgresfalse198‘
260616436RaffleTicket RaffleTicket_pkey
CONSTRAINTaALTER TABLE ONLY "RaffleTicket"
  ADD CONSTRAINT "RaffleTicket_pkey" PRIMARY KEY ("RaffleId");
LALTER TABLE ONLY public."RaffleTicket" DROP CONSTRAINT "RaffleTicket_pkey";
publicpostgresfalse200“
260616429 ScheduleEntry ScheduleEntry_pkey
CONSTRAINTeALTER TABLE ONLY "ScheduleEntry"

   ADD CONSTRAINT "ScheduleEntry_pkey" PRIMARY KEY ("ScheduleId");
NALTER TABLE ONLY public."ScheduleEntry" DROP CONSTRAINT "ScheduleEntry_pkey";
publicpostgresfalse199÷
260616502User User_pkey
CONSTRAINTOALTER TABLE ONLY "User"
   ADD CONSTRAINT "User_pkey" PRIMARY KEY ("UserId");
<ALTER TABLE ONLY public."User" DROP CONSTRAINT "User_pkey";
publicpostgresfalse202Œ
260616418VocalVote VocalVote_pkey
CONSTRAINT^ALTER TABLE ONLY "VocalVote"
   ADD CONSTRAINT "VocalVote_pkey" PRIMARY KEY ("VocalVoteId");
FALTER TABLE ONLY public."VocalVote" DROP CONSTRAINT "VocalVote_pkey";
publicpostgresfalse1973152.dat0000600 0004000 0002000 00000001007 13242700632 0014241
0ustar00postgrespostgres0000000 0000000 7	benitezb@ismanila.org
	Half Past Five
\.

3154.dat0000600 0004000 0002000 00000017051 13242700632 0014251
0ustar00postgrespostgres0000000 0000000 1	Black Tears
	Band
2	Cookie Problems
	Band
3	Twisted Adobo
	Band
4	Parental Advisory
	Band
5	The Dashboard Dolls
	Band
6	United States of Hysteria
	Band
7	University Parkway
	Band
8	Half Past Five
	Band
9	Silvana De Dios
	Vocalist
10	Meg Barraca & Kathleen Simba
	Vocalist
11	Margaret Nethercott
	Vocalist
12	Miguela Puyat
	Vocalist

13	Yojana Narang
	Vocalist
14	Iris Maloney
	Vocalist
15	Ben Reedy
	Vocalist
\.

3156.dat0000600 0004000 0002000 00000003051 13242700632 0014246
0ustar00postgrespostgres0000000 0000000 1	1	0	1
2	5	0	2
3	4	0	3
4	2	0	4
5	5	0	5
6	3	0	6
\.

3155.dat0000600 0004000 0002000 00000007553 13242700632 0014260
0ustar00postgrespostgres0000000 0000000 6	3	1	3
12	6	1	6
5	11	1	11
3	10	2	10
4	2	0	2
13	15	1	15
8	4	1	4
1	9	2	9
2	1	2	1
10	5	1	5
11	14	1	14
15	8	1	8
9	13	1	13
14	7	1	7
7	12	1	12
\.

3158.dat0000600 0004000 0002000 00000017041 13242700633 0014255
0ustar00postgrespostgres0000000 0000000 9	bettinabenitez16@gmail.com
	Bettina
	Benitez
	csisfun

Admin
2
10
John
Doe
whatsup
Admin
3
8 benitezb@ismanila.org
Bettina
Benitez
hello
Admin
1
11 bettinabenitez@gmail.com
Bettina
Benitez
hellooo
Admin
4
12 hoj@ismanila.org
Jackie
Ho
hello1
Admin
5
13 albeebenitez@gmail.com
albee
benitez
yoyoyo
Admin
6
\.

3153.dat0000600 0004000 0002000 00000001007 13242700633 0014243
0ustar00postgrespostgres0000000 0000000 5 benitezb@ismanila.org
Margaret Nethercott
\.

restore.sql0000600 0004000 0002000 00000017653 13242700633 0015400
0ustar00postgrespostgres0000000 0000000 --

```
-- NOTE:
--
-- File paths need to be edited. Search for $$PATH$$ and
-- replace it with the path to the directory containing
-- the extracted data files.
--
--
-- PostgreSQL database dump
--

-- Dumped from database version 10.1
-- Dumped by pg_dump version 10.1

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;

SET search_path = public, pg_catalog;

ALTER TABLE ONLY public."VocalVote" DROP CONSTRAINT "VocalVote_pkey";
ALTER TABLE ONLY public."User" DROP CONSTRAINT "User_pkey";
ALTER TABLE ONLY public."ScheduleEntry" DROP CONSTRAINT "ScheduleEntry_pkey";
ALTER TABLE ONLY public."RaffleTicket" DROP CONSTRAINT "RaffleTicket_pkey";
ALTER TABLE ONLY public."Performer" DROP CONSTRAINT "Performer_pkey";
ALTER TABLE ONLY public."BandVote" DROP CONSTRAINT "BandVote_pkey";
ALTER TABLE public."User" ALTER COLUMN "UserId" DROP DEFAULT;
DROP TABLE public."VocalVote";
DROP SEQUENCE public.vocalvote_id_seq;
DROP SEQUENCE public."User_UserId_seq";
DROP TABLE public."User";
DROP TABLE public."ScheduleEntry";
DROP TABLE public."RaffleTicket";
DROP TABLE public."Performer";
DROP TABLE public."BandVote";
DROP SEQUENCE public.bandvote_id_seq;
DROP EXTENSION plpgsql;
DROP SCHEMA public;
--
```

```
-- Name: public; Type: SCHEMA; Schema: -; Owner: postgres
--

CREATE SCHEMA public;


ALTER SCHEMA public OWNER TO postgres;

--
-- Name: SCHEMA public; Type: COMMENT; Schema: -; Owner: postgres
--

COMMENT ON SCHEMA public IS 'standard public schema';


--
-- Name: plpgsql; Type: EXTENSION; Schema: -; Owner:
--

CREATE EXTENSION IF NOT EXISTS plpgsql WITH SCHEMA pg_catalog;


--
-- Name: EXTENSION plpgsql; Type: COMMENT; Schema: -; Owner:
--

COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural language';


SET search_path = public, pg_catalog;

--
-- Name: bandvote_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE bandvote_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
```

```
ALTER TABLE bandvote_id_seq OWNER TO postgres;

SET default_tablespace = '';

SET default_with_oids = false;

--
-- Name: BandVote; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE "BandVote" (
   "BandVoteId" integer DEFAULT nextval('bandvote_id_seq'::regclass) NOT NULL,
   "User_Email" character(255),
   "Performer_Name" character(255)
);


ALTER TABLE "BandVote" OWNER TO postgres;

--
-- Name: Performer; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE "Performer" (
   "PerformerId" integer NOT NULL,
   "Performer_Name" character(255),
   "PerformerType" character(255)
);


ALTER TABLE "Performer" OWNER TO postgres;

--
-- Name: RaffleTicket; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE "RaffleTicket" (
   "RaffleId" integer NOT NULL,
   "UserId" integer,
   "Status" integer DEFAULT 0,
   "RaffleTicketNumber" character(255)
);
```

```
ALTER TABLE "RaffleTicket" OWNER TO postgres;

--
-- Name: ScheduleEntry; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE "ScheduleEntry" (
   "ScheduleId" integer NOT NULL,
   "PerformerId" integer,
   "Status" integer,
   "Perfomer_id" character(255)
);


ALTER TABLE "ScheduleEntry" OWNER TO postgres;

--
-- Name: User; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE "User" (
   "UserId" integer NOT NULL,
   "GmailAddress" character(255),
   "FirstName" character(255),
   "LastName" character(255),
   "Password" character(255),
   "UserType" character(255),
   "User_Id" integer
);


ALTER TABLE "User" OWNER TO postgres;

--
-- Name: User_UserId_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE "User_UserId_seq"
   AS integer
   START WITH 1
   INCREMENT BY 1
   NO MINVALUE
```

```
    NO MAXVALUE
    CACHE 1;


ALTER TABLE "User_UserId_seq" OWNER TO postgres;

--
-- Name: User_UserId_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--

ALTER SEQUENCE "User_UserId_seq" OWNED BY "User"."UserId";


--
-- Name: vocalvote_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE vocalvote_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE vocalvote_id_seq OWNER TO postgres;

--
-- Name: VocalVote; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE "VocalVote" (
    "VocalVoteId" integer DEFAULT nextval('vocalvote_id_seq'::regclass) NOT NULL,
    "User_Email" character(255),
    "Performer_Name" character(255)
);


ALTER TABLE "VocalVote" OWNER TO postgres;

--
-- Name: User UserId; Type: DEFAULT; Schema: public; Owner: postgres
--
```

```
ALTER TABLE ONLY "User" ALTER COLUMN "UserId" SET DEFAULT
nextval('"User_UserId_seq"'::regclass);


--
-- Data for Name: BandVote; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY "BandVote" ("BandVoteId", "User_Email", "Performer_Name") FROM stdin;
\.
COPY "BandVote" ("BandVoteId", "User_Email", "Performer_Name") FROM '$$PATH$$/3152.dat';

--
-- Data for Name: Performer; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY "Performer" ("PerformerId", "Performer_Name", "PerformerType") FROM stdin;
\.
COPY "Performer" ("PerformerId", "Performer_Name", "PerformerType") FROM '$$PATH$$/3154.dat';

--
-- Data for Name: RaffleTicket; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY "RaffleTicket" ("RaffleId", "UserId", "Status", "RaffleTicketNumber") FROM stdin;
\.
COPY "RaffleTicket" ("RaffleId", "UserId", "Status", "RaffleTicketNumber") FROM '$$PATH$$/3156.dat';

--
-- Data for Name: ScheduleEntry; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY "ScheduleEntry" ("ScheduleId", "PerformerId", "Status", "Perfomer_id") FROM stdin;
\.
COPY "ScheduleEntry" ("ScheduleId", "PerformerId", "Status", "Perfomer_id") FROM
'$$PATH$$/3155.dat';

--
-- Data for Name: User; Type: TABLE DATA; Schema: public; Owner: postgres
--
```

```
COPY "User" ("UserId", "GmailAddress", "FirstName", "LastName", "Password", "UserType", "User_Id")
FROM stdin;
\.
COPY "User" ("UserId", "GmailAddress", "FirstName", "LastName", "Password", "UserType", "User_Id")
FROM '$$PATH$$/3158.dat';

--
-- Data for Name: VocalVote; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY "VocalVote" ("VocalVoteId", "User_Email", "Performer_Name") FROM stdin;
\.
COPY "VocalVote" ("VocalVoteId", "User_Email", "Performer_Name") FROM '$$PATH$$/3153.dat';

--
-- Name: User_UserId_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres
--

SELECT pg_catalog.setval('"User_UserId_seq"', 13, true);


--
-- Name: bandvote_id_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres
--

SELECT pg_catalog.setval('bandvote_id_seq', 7, true);


--
-- Name: vocalvote_id_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres
--

SELECT pg_catalog.setval('vocalvote_id_seq', 5, true);


--
-- Name: BandVote BandVote_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY "BandVote"
   ADD CONSTRAINT "BandVote_pkey" PRIMARY KEY ("BandVoteId");
```

```
--
-- Name: Performer Performer_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY "Performer"
   ADD CONSTRAINT "Performer_pkey" PRIMARY KEY ("PerformerId");


--
-- Name: RaffleTicket RaffleTicket_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY "RaffleTicket"
   ADD CONSTRAINT "RaffleTicket_pkey" PRIMARY KEY ("RaffleId");


--
-- Name: ScheduleEntry ScheduleEntry_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY "ScheduleEntry"
   ADD CONSTRAINT "ScheduleEntry_pkey" PRIMARY KEY ("ScheduleId");


--
-- Name: User User_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY "User"
   ADD CONSTRAINT "User_pkey" PRIMARY KEY ("UserId");


--
-- Name: VocalVote VocalVote_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY "VocalVote"
   ADD CONSTRAINT "VocalVote_pkey" PRIMARY KEY ("VocalVoteId");


--
-- PostgreSQL database dump complete
--
```