
CS-223
SOFTWARE CODE REVIEW
DOCUMENT

for

Project 8
Classroom Visualisation App-2

Prepared by:

Group-2

Harshit Agrawal - 160101030

Harshit Gupta - 160101031

Harshit Sharma - 160101032

April 22, 2018

Contents

1	Introduction	3
1.1	Goals	3
2	Code	4
2.1	Activity	4
2.2	Add data	5
2.3	Display	7
2.4	Seat Status	10
2.5	Select Seat	15
2.6	Show Data	17
2.7	Layout	20
3	Code Testing Team	23
3.1	Team profile	23
4	Code Inspection Report	24
4.1	Code Report by Mitansh Jain	24
4.2	Code Report by Sujoy Ghosh	25
4.3	Code Report by Daman Tekchandani	26
5	Conclusion	27

1 Introduction

This document contains the complete software review of our app Classroom Visualisation App . Code review for a model is carried out after the module is successfully compiled and the all the syntax errors have been eliminated. Code reviews are extremely cost-effective strategies for reduction in coding errors and to produce high quality code. Normally, two types of reviews are carried out on the code of a module. These two types of code review techniques are code inspection and code walk through. The code testing team in isolation tests different units and modules of the system. Different members of the code testing team have submitted their reports. Although, we are only performing the Code Inspection part wherein each member goes through code to discover some common types of errors caused due to oversight and improper programming. The main objectives of the walk through was to discover the algorithmic and logical errors in the code. The members noted down their findings, which were discussed in a walk through meeting where the coder of the module were also present.

1.1 Goals

The main reasons for performing code review is to :

1. Finding bugs, since bug finding in code review are easier to find and fix, than later in testing.
2. Adherence to coding conventions
3. Improving code quality and understandability
4. Increasing efficiency, by finding trivial programming errors like data resource wastage or use of uninitialized variables.

2 Code

There are 7 modules in the code

2.1 Activity

```
Activity.java
1 package com.example.harshit.post;
2
3 /*
4 <header>
5 Module Name      : Activity
6 Date of Creation : 18-04-2018
7 Author          : Group 2
8
9 Modification History : 11-04-2018 :- added the activity of display classroom
10
11 Synopsis         : This is the main activity of the app
12
13 Global Variables : Button showData      button to show students record
14                  Button addData        button to add a student
15                  Button selectSeat     button to select a seat
16                  Button layoutGenerator button to generate classroom layout
17                  Button displayClassroom button to display classroom
18                  TextView networkStatus Text view to show the internet connectivity
19
20 Functions        : void addData        function to add a student
21                  void showData        function to show students record
22                  void selectSeat      function to select a seat
23                  void layoutGenerator function to generate classroom layout
24                  void displayClassroom function to display classroom
25 </header>
26 */
27
28 import android.content.Context;
29 import android.content.Intent;
30 import android.net.ConnectivityManager;
31 import android.net.NetworkInfo;
32 import android.support.v7.app.AppCompatActivity;
33 import android.os.Bundle;
34 import android.view.View;
35 import android.widget.Button;
36 import android.widget.TextView;
37
38 ...
39
40 public class Activity extends AppCompatActivity {
41     Button showData, addData, selectSeat, layoutGenerator, displayClassroom; // declaration of display buttons on the main screen .
42     TextView networkStatus; // declaration of text-view of network status
43
44     @Override
45     protected void onCreate(Bundle savedInstanceState) {
46         super.onCreate(savedInstanceState); // Calling the constructor of parent class.
47         setContentView(R.layout.activity_); // Setting the activity content to the main view.
48
49         // ****
50         // assigning the display button and text resource to the declared variable
51         showData = findViewById(R.id.showData);
52         addData = findViewById(R.id.addData);
53         selectSeat = findViewById(R.id.selectSeat);
54         layoutGenerator = findViewById(R.id.layoutGenerator);
55         displayClassroom = findViewById(R.id.displayClassroom);
56         networkStatus = findViewById(R.id.Network_status);
57         // ****
58
59         // checking the internet connectivity of the device
60         ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
61         NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo(); // fetching network information
62         if(networkInfo!=null && networkInfo.isConnected())
63         {
64             networkStatus.setVisibility(View.INVISIBLE);
65         }
66         else
67         {
68             // disabled buttons if internet is not connected
69             showData.setEnabled(false);
70             addData.setEnabled(false);
71             selectSeat.setEnabled(false);
72             layoutGenerator.setEnabled(false);
73             displayClassroom.setEnabled(false);
74         }
75     }
76 }
```

```

50         displayLayoutClassroom = findViewById(R.id.displayLayoutClassroom);
51         networkStatus = findViewById(R.id.Network_status);
52         // ****
53
54         // checking the internet connectivity of the device
55         ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
56         NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo(); // fetching network information
57
58         if(networkInfo!=null &&networkInfo.isConnected())
59         {
60             networkStatus.setVisibility(View.INVISIBLE);
61         }
62         else
63         {
64             // disabled buttons if internet is not connected
65             showData.setEnabled(false);
66             addData.setEnabled(false);
67             selectSeat.setEnabled(false);
68             displayClassroom.setEnabled(false);
69             layoutGenerator.setEnabled(false);
70         }
71     }
72
73     //****
74     // Declaring the on-click screens(activity) of the display buttons
75     public void addData(View view)
76     {
77         startActivity(new Intent(this,addData.class));
78     }
79     public void showData(View view)
80     {
81         startActivity(new Intent(this,showData.class));
82     }
83     public void selectSeat(View view) {startActivity(new Intent(this,selectSeat.class));}
84     public void layoutGenerator(View view) {startActivity(new Intent(this,Layout.class));}
85     public void displayClassroom(View view) {startActivity(new Intent(this,display.class));}
86     //****
87 }

```

2.2 Add data

```

addData.java
1 package com.example.harshit.post;
2
3 /*
4  <header>
5  Module Name      : addData
6  Date of Creation : 18-04-2018
7  Author          : Group 2
8
9  Modification History : 11-04-2018 :- added exception for negative student id.
10
11 Synopsis          : This module is responsible for adding a student in the record
12
13 Global Variables  : EditText ID      Input from the user for student id
14                   : EditText Name    Input from the user for student name
15                   : String Student ID, Variable to send student id to database
16                   : String StudentName Variable to send student name to database
17
18 Functions          : void saveInfo    save the student id and the student name to the database
19  </header>
20  */
21
22 import android.os.AsyncTask;
23 import android.os.Bundle;
24 import android.support.v7.app.AppCompatActivity;
25 import android.view.View;
26 import android.widget.EditText;
27 import android.widget.Toast;
28
29 import java.io.BufferedWriter;
30 import java.io.IOException;
31 import java.io.InputStream;
32 import java.io.OutputStream;
33 import java.io.OutputStreamWriter;
34 import java.net.HttpURLConnection;
35 import java.net.MalformedURLException;
36 import java.net.URL;
37 import java.net.URLEncoder;
38

```

```

37 import java.net.URLEncoder;
38
39 public class addData extends AppCompatActivity {
40
41     EditText ID,Name; // declaration of text input on the add data screen .
42     String Student_ID,StudentName; // declaration of Student ID , Student Name .
43
44     @Override
45     protected void onCreate(Bundle savedInstanceState) {
46
47         super.onCreate(savedInstanceState); // Calling the constructor of parent class.
48         setContentView(R.layout.activity_add_data); // Setting the activity content to the add data view.
49
50         /**
51          * Get the widgets reference from XML layout
52          * assigning the display button and text resource to the declared variable
53          */
54         ID = findViewById(R.id.ID);
55         Name = findViewById(R.id.Name);
56         /**
57          */
58     }
59
60     public void saveInfo(View view){
61         // fetching the id and the name from the user input
62         Student_ID = ID.getText().toString();
63         StudentName = Name.getText().toString();
64         // Starting the add request to the online database server
65         BackgroundTask backgroundTask = new BackgroundTask();
66         backgroundTask.execute(Student_ID,StudentName);
67         finish(); // exit the current screen when data is added .
68     }
69
70     class BackgroundTask extends AsyncTask<String, Void, String> {
71
72         String ADD_DATA_URL; // url of the database request handling script
73         @Override
74         protected void onPreExecute(){ // task to be performed before execution of the request to the server
75
76             ADD_DATA_URL = "https://22harshit.000webhostapp.com/add_info.php";
77
78         }
79
80         @Override
81         protected String doInBackground(String... args) {
82             String id, name; // declaring the id and the name from the arguments given to the function
83             // fetching the id and the name from the arguments given to the function
84             id = args[0];
85             name = args[1];
86             if (Integer.valueOf(id) >= 0){
87                 try {
88                     //creating a URL
89                     URL url = new URL(ADD_DATA_URL);
90                     //Opening the URL using HttpURLConnection
91                     HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
92                     // specifying the request method
93                     httpURLConnection.setRequestMethod("POST");
94                     httpURLConnection.setDoOutput(true);
95                     // creating a output stream
96                     OutputStream outputStream = httpURLConnection.getOutputStream();
97                     //We will use a buffered reader to read the string from service
98                     BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream, "UTF-8"));
99                     // encoding the data string to be sent
100                     String data_string = URLEncoder.encode("ID", "UTF-8") + "=" + URLEncoder.encode(id, "UTF-8") + "&" +
101                         URLEncoder.encode("Name", "UTF-8") + "=" + URLEncoder.encode(name, "UTF-8");
102                     // writing the data to the buffer
103                     bufferedWriter.write(data_string);
104
105                     // closing and flushing the connections
106                     bufferedWriter.flush();
107                     bufferedWriter.close();
108                     outputStream.close();
109                     InputStream inputStream = httpURLConnection.getInputStream();
110                     inputStream.close();
111                     httpURLConnection.disconnect();
112                     //finally returning the status string
113                     return "Data was inserted";

```

```

addData.java - Google Chrome
109         InputStream inputStream = httpURLConnection.getInputStream();
110         inputStream.close();
111         httpURLConnection.disconnect();
112         //finally returning the status string
113         return "One row data inserted";
114     }
115     } catch (MalformedURLException e) {
116         e.printStackTrace();
117     } catch (IOException e) {
118         e.printStackTrace();
119     }
120     return null;
121 }
122 else
123 {
124     return "Error : negative Student ID";
125 }
126 }
127
128 @Override
129 protected void onProgressUpdate(Void ... values){
130     super.onProgressUpdate(values);
131 }
132
133 @Override
134 protected void onPostExecute(String result)
135 {
136     // printing the success message
137     Toast.makeText(getApplicationContext(),result,Toast.LENGTH_LONG).show();
138 }
139
140 }
141
142 }
143
144
145
146

```

2.3 Display

```

display.java
1 package com.example.harshit.post;
2
3 /*
4 <header>
5 Module Name      : display
6 Date of Creation : 10-04-2018
7 Author          : Group 2
8
9 Modification History : 11-04-2018 :- added try catch statements to handle crashing of the app
10
11 Synopsis          : This module is responsible for displaying the classroom
12
13 Global Variables   : GridView GridView  gridview to represent seats
14
15 Functions          : void getJSON      get seat information in json format from database
16                    : void loadIntoListView load the json format array to list view to display
17 </header>
18 */
19
20
21 import android.graphics.Color;
22 import android.os.AsyncTask;
23 import android.support.v7.app.AppCompatActivity;
24 import android.os.Bundle;
25 import android.util.Log;
26 import android.view.ViewTreeObserver;
27 import android.widget.AdapterView;
28 import android.widget.AdapterView.OnItemClickListener;
29 import android.widget.GridView;
30 import android.widget.TextView;
31 import android.widget.Toast;
32
33 import org.json.JSONArray;
34 import org.json.JSONException;
35 import org.json.JSONObject;
36
37 import java.io.BufferedReader;
38 import java.io.InputStreamReader;
39

```

```

34 import org.json.JSONObject;
35
36 import java.io.BufferedReader;
37 import java.io.InputStreamReader;
38 import java.net.HttpURLConnection;
39 import java.net.URL;
40
41 public class display extends AppCompatActivity {
42
43     GridView GridView;
44
45     @Override
46     protected void onCreate(Bundle savedInstanceState) {
47         super.onCreate(savedInstanceState); // Calling the constructor of parent class.
48         setContentView(R.layout.activity_display); // Setting the activity content to the display view
49
50         // Get the widgets reference from XML layout
51         // assigning the display button and text resource to the declared variable
52         GridView = (GridView)findViewById(R.id.display);
53
54         getJSON("https://22harshit.00webhostapp.com/display.php"); // Fetching database entries of the seats information database
55     }
56
57     private void getJSON(final String urlWebService) {
58         /*
59          * As fetching the json string is a network operation
60          * And we cannot perform a network operation in main thread
61          * so we need an AsyncTask
62          * The constraints defined here are
63          * Void -> We are not passing anything
64          * Void -> Nothing at progress update as well
65          * String -> After completion it should return a string and it will be the json string
66          */
67         class GetJSON extends AsyncTask<Void, Void, String> {
68
69             //this method will be called before execution
70             //you can display a progress bar or something
71             //so that user can understand that he should wait
72             //as network operation may take some time
73             @Override
74
75             //you can display a progress bar or something
76             //so that user can understand that he should wait
77             //as network operation may take some time
78             @Override
79             protected void onPreExecute() {
80                 super.onPreExecute();
81             }
82
83             //this method will be called after execution
84             //so here we are displaying a toast with the json string
85             @Override
86             protected void onPostExecute(String s) {
87                 super.onPostExecute(s);
88                 if(s!=null) {
89                     try {
90                         loadIntoListView(s);
91                     } catch (JSONException e) { // try catch exception for JSON
92                         e.printStackTrace();
93                     }
94                 }
95                 else
96                 {
97                     Toast.makeText(getApplicationContext(), "Internet not working", Toast.LENGTH_LONG).show();
98                 }
99             }
100
101             //in this method we are fetching the json string
102             @Override
103             protected String doInBackground(Void... voids) {
104
105                 try {
106                     //creating a URL
107                     URL url = new URL(urlWebService);
108
109                     //Opening the URL using HttpURLConnection
110                     HttpURLConnection con = (HttpURLConnection) url.openConnection();
111
112                     //StringBuilder object to read the string from the service
113                     StringBuilder sb = new StringBuilder();

```



```

108 //StringBuilder object to read the string from the service
109 StringBuilder sb = new StringBuilder();
110
111 //We will use a buffered reader to read the string from service
112 BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(con.getInputStream()));
113
114 //A simple string to read values from each line
115 String json;
116
117 //reading until we don't find null
118 while ((json = bufferedReader.readLine()) != null) {
119
120     //appending it to string builder
121     sb.append(json).append("\n");
122 }
123
124 //finally returning the read string
125 return sb.toString().trim();
126 } catch (Exception e) {
127     return null;
128 }
129 }
130 }
131
132 //creating async task object and executing it
133 GetJSON getJSON = new GetJSON();
134 getJSON.execute();
135 }
136 }
137
138 private void loadIntoListView(String json) throws JSONException {
139     //creating a json array from the json string
140     final JSONArray jsonArray = new JSONArray(json);
141
142     //creating a string array for listview
143     String[] SeatsInfo = new String[jsonArray.length()];
144
145     //looping through all the elements in json array

```

```

144
145 //looping through all the elements in json array
146 for (int i = 0; i < jsonArray.length(); i++) {
147
148     //getting json object from the json array
149     JSONObject seat = jsonArray.getJSONObject(i);
150     //getting the name from the json object and putting it inside string array
151     SeatsInfo[i] = seat.getString("seat_number") + " : " + seat.getString("student_ID");
152     Log.i("data", SeatsInfo[i]);
153 }
154
155 //the array adapter to load data into list
156 ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, SeatsInfo);
157
158 //attaching adapter to listview
159 GridView.setAdapter(arrayAdapter);
160
161 GridView.getViewTreeObserver().addOnGlobalLayoutListener(
162     new ViewTreeObserver.OnGlobalLayoutListener() {
163         @Override
164         public void onGlobalLayout() {
165             final int size = GridView.getChildCount();
166             for(int i = 0; i < size; i++) {
167
168                 TextView gridChild = (TextView) GridView.getChildAt(i);
169                 try {
170                     if(Integer.parseInt(jsonArray.getJSONObject(i).getString("status")) == 0) {
171                         gridChild.setBackgroundColor(Color.GRAY);
172                     }
173                     else
174                     {
175                         // fetching the state of the student of the selected seat
176                         int state = Integer.parseInt(jsonArray.getJSONObject(i).getString("state"));
177                         // assigning color codes to the seats based on the state value
178                         if(state<=4)
179                         {
180                             gridChild.setBackgroundColor(Color.RED);
181                         }
182                         // state value is between 5 and 7

```

```

174 if(Integer.parseInt(jsonArray.getJSONObject(2).getString("status")) == 0) {
175     gridChild.setBackgroundColor(Color.GRAY);
176 }
177 else
178 {
179     // fetching the state of the student of the selected seat
180     int state = Integer.parseInt(jsonArray.getJSONObject(1).getString("state"));
181     // assigning color codes to the seats based on the state value
182     if(state==4)
183     {
184         gridChild.setBackgroundColor(Color.RED);
185     }
186     else if(state==7) // for state value in between 5 and 7
187     {
188         gridChild.setBackgroundColor(Color.BLUE);
189     }
190     else // i.e for state values >7
191     {
192         gridChild.setBackgroundColor(Color.GREEN);
193     }
194 }
195 } catch (JSONException e) { // avoid error if Json array is NULL
196     e.printStackTrace();
197 }
198 }
199 }
200 };
201 }
202 }
203 }
204 }
205 }
206 }
207 }

```

2.4 Seat Status

```

1 package com.example.harshit.post;
2
3 /*
4  <header>
5  Module Name      : seatStatus
6  Date of Creation : 18-04-2018
7  Author          : Group 2
8
9  Modification History : 11-04-2018 :- added colour to the last seat selected
10
11 Synopsis          : This module is responsible for selecting unoccupied seats by
12                    checking their status.
13
14 Global Variables  :      TextView ID          declaration of text view on the seatStatus screen .
15                    Gridview GridView        declaration of Seat Grid View on the seatStatus screen .
16                    Button ConfirmSeat        declaration of Confirm button on the seatStatus screen .
17                    String SeatNumber         variable to send seat no. to the database
18                    String StudentID         variable to send student id. to the database
19                    String SelectedSeat       variable to send seat no. of the seat selected to the database
20                    int previousSelected      Position represents the last seat selected
21
22 Functions         :      void saveInfo        function that saves the information in the database
23                    void onItemSelected      function that fetches the seat no. of the selected seat
24                    void getJSON            get seat information in json format from database
25                    void loadIntoListView    load the json format array to list view to display
26  </header>
27  */
28
29 import android.graphics.Color;
30 import android.os.AsyncTask;
31 import android.support.v7.app.AppCompatActivity;
32 import android.os.Bundle;
33 import android.view.View;
34 import android.widget.AdapterView;
35 import android.widget.AdapterView.OnItemClickListener;
36 import android.widget.Button;
37 import android.widget.GridView;

```

```

37 import android.widget.GridView;
38 import android.widget.TextView;
39 import android.widget.Toast;
40
41 import org.json.JSONArray;
42 import org.json.JSONException;
43 import org.json.JSONObject;
44
45 import java.io.BufferedReader;
46 import java.io.BufferedWriter;
47 import java.io.IOException;
48 import java.io.InputStream;
49 import java.io.InputStreamReader;
50 import java.io.OutputStream;
51 import java.io.OutputStreamWriter;
52 import java.net.HttpURLConnection;
53 import java.net.MalformedURLException;
54 import java.net.URL;
55 import java.net.URLEncoder;
56 import java.util.Random;
57
58 public class seatStatus extends AppCompatActivity implements AdapterView.OnItemClickListener {
59
60     TextView ID; // declaration of text view on the seatStatus screen .
61     GridView GridView; // declaration of Seat Grid View on the seatStatus screen .
62     Button ConfirmSeat; // declaration of Confirm button on the seatStatus screen .
63     String SeatNumber;
64     String StudentID="";
65     String SelectedSeat;
66     int previousSelectedPosition=-1;
67
68     @Override
69     protected void onCreate(Bundle savedInstanceState) {
70         super.onCreate(savedInstanceState); // Calling the constructor of parent class.
71         setContentView(R.layout.activity_seat_status); // Setting the activity content to the add data view.
72
73         // Get the widgets reference from XML layout
74         // assigning the display button and text resource to the declared variable
75         GridView = (GridView) findViewById(R.id.gridLayout);
76         ID = (TextView) findViewById(R.id.Sudent_ID);
77
78
79
80         ID = (TextView) findViewById(R.id.Sudent_ID);
81         ConfirmSeat = (Button) findViewById(R.id.confirm_seat);
82
83         // extracting the information passed on by previous activity
84         Bundle bundle = getIntent().getExtras();
85         ID.setText(bundle.getString("ID"));
86         StudentID = bundle.getString("ID");
87
88         // Fetching database entries of the seats available
89         getJSON("https://22harshit.000webhostapp.com/out_seats.php");
90
91     }
92
93     public void saveInfo(View view){
94         // fetching the id and the seat number from the user input
95         StudentID = ID.getText().toString();
96         SeatNumber = SelectedSeat;
97         BackgroundTask backgroundTask = new BackgroundTask();
98         // Starting the add request to the online database server
99         backgroundTask.execute(StudentID,SeatNumber);
100         finish(); // exit the current screen when data is added .
101     }
102
103     public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
104         // On selecting a spinner item
105         SelectedSeat = parent.getItemAtPosition(position).toString();
106
107         // Showing selected spinner item
108         Toast.makeText(parent.getContext(), "Selected: " + SelectedSeat, Toast.LENGTH_LONG).show();
109     }
110
111     public void onNothingSelected(AdapterView<?> arg0) {
112
113     }
114
115     private void getJSON(final String urlWebService) {
116         /*
117          * As fetching the json string is a network operation
118          */
119     }
120

```

```

112 private void getJSON(final String urlWebService) {
113     /*
114     * As fetching the json string is a network operation
115     * And we cannot perform a network operation in main thread
116     * so we need an AsyncTask
117     * The constraints defined here are
118     * Void -> We are not passing anything
119     * Void -> Nothing at progress update as well
120     * String -> After completion it should return a string and it will be the json string
121     */
122     class GetJSON extends AsyncTask<Void, Void, String> {
123
124         //this method will be called before execution
125         //you can display a progress bar or something
126         //so that user can understand that he should wait
127         //as network operation may take some time
128         @Override
129         protected void onPreExecute() {
130             super.onPreExecute();
131         }
132
133         //this method will be called after execution
134         //so here we are displaying a toast with the json string
135         @Override
136         protected void onPostExecute(String s) {
137             super.onPostExecute(s);
138
139             try {
140                 loadIntoListView(s);
141             } catch (JSONException e) {
142                 e.printStackTrace();
143             }
144         }
145     }
146
147     //in this method we are fetching the json string
148     @Override
149     protected String doInBackground(Void... voids) {
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

187         getJSON.execute();
188     }
189
190     private void loadIntoListView(String json) throws JSONException {
191         //creating a json array from the json string
192         JSONArray jsonArray = new JSONArray(json);
193
194         //creating a string array for listview
195         String[] seatStatus = new String[jArray.length()];
196
197         //looping through all the elements in json array
198         for (int i = 0; i < jsonArray.length(); i++) {
199             //getting json object from the json array
200             JSONObject obj = jsonArray.getJSONObject(i);
201             //getting the name from the json object and putting it inside string array
202             seatStatus[i] = obj.getString("seat_number");
203         }
204
205         //the array adapter to load data into list
206         ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, seatStatus);
207
208         //attaching adapter to list view
209         GridView.setAdapter(arrayAdapter);
210         GridView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
211
212             @Override
213             public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
214                 String selectedItem = parent.getItemAtPosition(position).toString();
215
216                 // Get the current selected view as a TextView
217                 TextView tv = (TextView) view;
218
219                 // Set the current selected item background color
220                 tv.setBackgroundColor(Color.parseColor("#FF9AD082"));
221
222                 // Set the current selected item text color
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266

```

```

// Set the current selected item text color
tv.setTextColor(Color.BLUE);

// Get the last selected View from GridView
TextView previousSelectedView = (TextView) GridView.getChildAt(previousSelectedPosition);
SelectedSeat=Integer.toString(position+1);
// If there is a previous selected view exists
if (previousSelectedPosition != -1)
{
    // Set the last selected View to deselect
    previousSelectedView.setSelected(false);

    // Set the last selected View background color as deselected item
    previousSelectedView.setBackgroundColor(Color.parseColor("#FFFFFF"));

    // Set the last selected View text color as deselected item
    previousSelectedView.setTextColor(Color.DKGRAY);
}

// Set the current selected view position as previousSelectedPosition
previousSelectedPosition = position;
}
});
}

class BackgroundTask extends AsyncTask<String, Void, String> {
    String SELECT_SEAT_URL;
    @Override
    protected void onPreExecute(){
        SELECT_SEAT_URL = "http://22harshit.000webhostapp.com/new_seat.php";
    }

    @Override
    protected String doInBackground(String... args) {

```

```

257 String SELECT_SEAT_URL;
258
259 @Override
260 protected void onPreExecute(){
261     SELECT_SEAT_URL = "http://22harshit.000webhostapp.com/new_seat.php";
262 }
263
264
265
266 @Override
267 String doInBackground(String... args) {
268     String id,seat_number;
269     Random rnd = new Random();
270     String state = Integer.toString(rnd.nextInt(10) + 1);
271     id = args[0];
272     seat_number = args[1];
273     try {
274         //creating a URL
275         URL url = new URL(SELECT_SEAT_URL);
276         //Opening the URL using HttpURLConnection
277         HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
278         // specifying the request method
279         httpURLConnection.setRequestMethod("POST");
280         httpURLConnection.setDoOutput(true);
281         // creating a output stream
282         OutputStream outputStream = httpURLConnection.getOutputStream();
283         //We will use a buffered reader to read the string from service
284         BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream,"UTF-8"));
285         // encoding the data string to be sent
286         String data_string = URLEncoder.encode("ID","UTF-8")+"="+URLEncoder.encode(id,"UTF-8")+"&"+
287             URLEncoder.encode("seat_number","UTF-8")+"="+URLEncoder.encode(seat_number,"UTF-8")+"&"+
288             URLEncoder.encode("state","UTF-8")+"="+URLEncoder.encode(state,"UTF-8");
289         // writing the data to the buffer
290         bufferedWriter.write(data_string);
291
292         // closing and flushing the connections
293         bufferedWriter.flush();
294         bufferedWriter.close();
295         outputStream.close();
296
297         // reading the response from server
298
299         // reading the response from server
300         InputStream inputStream = httpURLConnection.getInputStream();
301         BufferedReader r = new BufferedReader(new InputStreamReader(inputStream));
302         StringBuilder statusMessage = new StringBuilder();
303         String line;
304         while ((line = r.readLine()) != null) {
305             statusMessage.append(line).append('\n');
306         }
307         inputStream.close();
308         httpURLConnection.disconnect();
309
310         //finally returning the status string
311         return statusMessage.toString();
312     } catch (MalformedURLException e) {
313         e.printStackTrace();
314     } catch (IOException e) {
315         e.printStackTrace();
316     }
317     return null;
318 }
319
320 @Override
321 protected void onProgressUpdate(Void ... values){
322     super.onProgressUpdate(values);
323 }
324
325 @Override
326 protected void onPostExecute(String result)
327 {
328     // printing the status message
329     Toast.makeText(getApplicationContext(),result,Toast.LENGTH_LONG).show();
330 }
331
332 }
333

```

2.5 Select Seat

```
selectSeat.java
1 package com.example.harshit.post;
2
3 /*
4 <header>
5 Module Name      : selectSeat
6 Date of Creation : 10-04-2018
7 Author          : Group 2
8
9 Modification History : 11-04-2018 :- added the exception for existing id allocated seat
10
11 Synopsis         : This module is responsible for checking the student id and proceed for seat selection
12
13 Global Variables : EditText ID      declaration of text input on the select seat screen .
14                  String StudentID  declaration of Student ID .
15                  Button selectSeat button to go for seat selection
16
17 Functions        : void checkStudent check if the student is enrolled in the course or not
18                  void seatStatus   fetches seat status and proceed for seat selection
19 </header>
20 */
21
22 import android.content.Intent;
23 import android.os.AsyncTask;
24 import android.support.v7.app.AppCompatActivity;
25 import android.os.Bundle;
26 import android.util.Log;
27 import android.view.View;
28 import android.widget.Button;
29 import android.widget.EditText;
30 import android.widget.Toast;
31
32 import java.io.BufferedReader;
33 import java.io.BufferedWriter;
34 import java.io.IOException;
35 import java.io.InputStream;
36 import java.io.InputStreamReader;
37 import java.io.OutputStream;
38
39 import java.io.OutputStreamWriter;
40 import java.net.HttpURLConnection;
41 import java.net.MalformedURLException;
42 import java.net.URL;
43 import java.net.URLEncoder;
44
45 import static java.lang.Boolean.FALSE;
46 import static java.lang.Boolean.TRUE;
47
48 public class selectSeat extends AppCompatActivity {
49     EditText ID; // declaration of text input on the select seat screen .
50     String StudentID; // declaration of Student ID .
51     Button selectSeat; // declaration of select seat button on the select seat screen . .
52
53     @Override
54     protected void onCreate(Bundle savedInstanceState) {
55         super.onCreate(savedInstanceState); // Calling the constructor of parent class.
56         setContentView(R.layout.activity_select_seat); // Setting the activity content to the select seat view
57
58         // Get the widgets reference from XML layout
59         // assigning the display button and text resource to the declared variable
60         ID = (EditText) findViewById(R.id.StudentID);
61         selectSeat = (Button) findViewById(R.id.selectSeat);
62     }
63
64     public void checkStudent(View view) {
65         // fetching the id from the user input
66         StudentID = ID.getText().toString();
67         BackgroundTask backgroundTask = new BackgroundTask();
68         backgroundTask.execute(StudentID);
69     }
70
71     class BackgroundTask extends AsyncTask<String, Void, String> {
72         String CHECK_INFO_URL;
73
74         @Override
75         protected void onPreExecute() {
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
```

```

76     protected void onPreExecute() {
77         CHECK_INFO_URL = "https://22harshit.000webhostapp.com/check.php";
78     }
79
80
81
82
83     @Override
84     protected String doInBackground(String... args) {
85         String id;
86         id = args[0];
87         try {
88             //creating a URL
89             URL url = new URL(CHECK_INFO_URL);
90             //Opening the URL using HttpURLConnection
91             HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
92             httpURLConnection.setRequestMethod("POST");
93             httpURLConnection.setDoOutput(true);
94             // creating a output stream
95             OutputStream outputStream = httpURLConnection.getOutputStream();
96             //We will use a buffered reader to read the string from service
97             BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream, "UTF-8"));
98             // encoding the data string to be sent
99             String data_string = URLEncoder.encode("ID", "UTF-8") + "=" + URLEncoder.encode(id, "UTF-8");
100             // writing the data to the buffer
101             bufferedWriter.write(data_string);
102
103             // closing and flushing the connections
104             bufferedWriter.flush();
105             bufferedWriter.close();
106             bufferedWriter.close();
107             outputStream.close();
108
109             // reading the response from server
110             InputStream inputStream = httpURLConnection.getInputStream();
111             BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
112             String line;
113             String statusMessage = new StringBuilder();
114             while ((line = bufferedReader.readLine()) != null) {
115                 statusMessage.append(line).append('\n');
116             }
117         } catch (Exception e) {
118             e.printStackTrace();
119         }
120         return statusMessage.toString();
121     }
122
123     } catch (MalformedURLException e) {
124         e.printStackTrace();
125     } catch (IOException e) {
126         e.printStackTrace();
127     }
128     }
129     return null;
130 }
131
132 @Override
133 protected void onProgressUpdate(Void... values) {
134     super.onProgressUpdate(values);
135 }
136
137 @Override
138 protected void onPostExecute(String result) {
139     super.onPostExecute(result);
140
141     // trimming of the data to convert to integer
142     if(result!=null) {
143         result = result.replaceAll("\s", "");
144         int status = Integer.parseInt(result.replaceAll("[\\D]", ""));
145         // checking whether the student id is in the database or not
146         if (status == 1) {
147             selectSeat.setEnabled(TRUE);
148         } else {
149             // printing the status message
150             Toast.makeText(getApplicationContext(), "Enter a valid Student ID", Toast.LENGTH_LONG).show();
151             selectSeat.setEnabled(FALSE);
152         }
153     }
154 }

```

```

112     StringBuilder statusMessage = new StringBuilder();
113     String line;
114     while ((line = bufferedReader.readLine()) != null) {
115         statusMessage.append(line).append('\n');
116     }
117     inputStream.close();
118     httpURLConnection.disconnect();
119     //finally returning the status string
120     return statusMessage.toString();
121 }
122
123 } catch (MalformedURLException e) {
124     e.printStackTrace();
125 } catch (IOException e) {
126     e.printStackTrace();
127 }
128 }
129 return null;
130 }
131
132 @Override
133 protected void onProgressUpdate(Void... values) {
134     super.onProgressUpdate(values);
135 }
136
137 @Override
138 protected void onPostExecute(String result) {
139     super.onPostExecute(result);
140
141     // trimming of the data to convert to integer
142     if(result!=null) {
143         result = result.replaceAll("\s", "");
144         int status = Integer.parseInt(result.replaceAll("[\\D]", ""));
145         // checking whether the student id is in the database or not
146         if (status == 1) {
147             selectSeat.setEnabled(TRUE);
148         } else {
149             // printing the status message
150             Toast.makeText(getApplicationContext(), "Enter a valid Student ID", Toast.LENGTH_LONG).show();
151             selectSeat.setEnabled(FALSE);
152         }
153     }
154 }

```



```

132         super.onProgressUpdate(values);
133     }
134 }
135
136 @Override
137 protected void onPostExecute(String result) {
138     super.onPostExecute(result);
139
140     // trimming of the data to convert to integer
141     if(result!=null) {
142         result = result.replaceAll("\\s", "");
143         int status = Integer.parseInt(result.replaceAll("[\\D]", ""));
144         // checking whether the student id is in the database or not
145         if (status == 1) {
146             selectSeat.setEnabled(TRUE);
147         } else {
148             // printing the status message
149             Toast.makeText(getApplicationContext(), "Enter a valid Student ID", Toast.LENGTH_LONG).show();
150             selectSeat.setEnabled(FALSE);
151         }
152     }
153 }
154 else
155     Toast.makeText(getApplicationContext(), "Internet not working", Toast.LENGTH_LONG).show();
156 }
157
158 }
159
160
161 public void seatStatus(View view){ // function displaying available seats for the students
162     Bundle bundle = new Bundle(); // creating a information bundle to pass data to the next view
163     bundle.putString("ID",ID.getText().toString()); // adding information to the bundle
164     Intent intent = new Intent(this,seatStatus.class); // creating a new intent
165     intent.putExtras(bundle); // attaching the extra information with the intent
166     startActivity(intent); // starting a new intent ( screen )
167 }
168
169 }
170
171
172

```

2.6 Show Data

```

showData.java
1 package com.example.harshit.post;
2
3 /*
4 <header>
5 Module Name      : showData
6 Date of Creation  : 18-04-2018
7 Author           : Group 2
8
9 Modification History : 11-04-2018 :- added try catch statements for error handling
10
11 Synopsis          : This module is responsible for showing student record.
12
13 Global Variables   : ListView listView      - View to show the existing students enrolled in the course
14
15
16 Functions          : void getJSON          get student information in json format from database
17                   : void loadIntoListView  load the json format array to list view to display student record
18 </header>
19 */
20
21 import android.os.AsyncTask;
22 import android.support.v7.app.AppCompatActivity;
23 import android.os.Bundle;
24 import android.util.Log;
25 import android.view.View;
26 import android.widget.AdapterView;
27 import android.widget.AdapterView;
28 import android.widget.Toast;
29
30 import org.json.JSONArray;
31 import org.json.JSONException;
32 import org.json.JSONObject;
33
34 import java.io.BufferedReader;
35 import java.io.InputStreamReader;
36 import java.net.HttpURLConnection;
37 import java.net.URL;

```

```

37 import java.net.URL;
38 import java.sql.Connection;
39 import java.sql.DriverManager;
40 import java.sql.ResultSet;
41 import java.sql.Statement;
42
43 public class showData extends AppCompatActivity {
44
45     ListView listView ; // declaration of list view on the show data screen .
46
47     @Override
48     protected void onCreate(Bundle savedInstanceState) {
49         setContentView(R.layout.activity_show_data); // Setting the activity content to the add data view.
50         super.onCreate(savedInstanceState); // Calling the constructor of parent class
51         listView = findViewById(R.id.student_list); // assigning the list view resource to the declared variable
52         getJSON("https://22harshit.000webhostapp.com/out.php"); // Fetching database entries of the students enrolled
53     }
54
55     private void getJSON(final String urlWebService) {
56         /*
57          * As fetching the json string is a network operation
58          * And we cannot perform a network operation in main thread
59          * so we need an AsyncTask
60          * The constraints defined here are
61          * Void -> We are not passing anything
62          * Void -> Nothing at progress update as well
63          * String -> After completion it should return a string and it will be the json string
64          */
65         class GetJSON extends AsyncTask<Void, Void, String> {
66
67             //this method will be called before execution
68             //you can display a progress bar or something
69             //so that user can understand that he should wait
70             //as network operation may take some time
71             @Override
72             protected void onPreExecute() {
73                 super.onPreExecute();
74             }
75
76             //this method will be called after execution
77
78             super.onPreExecute();
79         }
80
81         //this method will be called after execution
82         //so here we are displaying a toast with the json string
83         @Override
84         protected void onPostExecute(String s) {
85             super.onPostExecute(s);
86             try {
87                 if(s!=null)
88                 {
89                     loadIntoListView(s);
90                 }
91                 else
92                 {
93                     Toast.makeText(getApplicationContext(), "Internet connection is not working ", Toast.LENGTH_LONG).show();
94                 }
95             } catch (JSONException e) {
96                 e.printStackTrace();
97             }
98         }
99
100         //in this method we are fetching the json string
101         @Override
102         protected String doInBackground(Void... voids) {
103
104             try {
105                 //creating a URL
106                 URL url = new URL(urlWebService);
107
108                 //Opening the URL using HttpURLConnection
109                 HttpURLConnection con = (HttpURLConnection) url.openConnection();
110
111                 //StringBuilder object to read the string from the service
112                 StringBuilder sb = new StringBuilder();
113
114                 //We will use a buffered reader to read the string from service
115                 BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(con.getInputStream()));
116
117                 //A simple string to read values from each line

```

```

112         //A simple string to read values from each line
113         String json;
114
115         //reading until we don't find null
116         while ((json = bufferedReader.readLine()) != null) {
117             //appending it to string builder
118             sb.append(json).append("\n");
119         }
120
121         //finally returning the read string
122         return sb.toString().trim();
123     } catch (Exception e) {
124         return null;
125     }
126 }
127
128 }
129
130 }
131
132 //creating async task object and executing it
133 GetJSON getJSON = new GetJSON();
134 getJSON.execute();
135 }
136
137 private void loadIntoListView(String json) throws JSONException {
138     //creating a json array from the json string
139     JSONArray jsonArray = new JSONArray(json);
140
141     //creating a string array for listview
142     String[] studentdataList = new String[jsonArray.length()];
143
144     //looping through all the elements in json array
145     for (int i = 0; i < jsonArray.length(); i++) {
146         //getting json object from the json array
147         JSONObject student = jsonArray.getJSONObject(i);
148         //getting the name from the json object and putting it inside string array
149         studentdataList[i] = student.getString("ID") + " : " + student.getString("Name");
150         Log.i("data", studentdataList[i]);
151     }
152 }
153
154 }
155
156 //the array adapter to load data into list
157 ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, studentdataList);
158
159 //attaching adapter to listview
160 Log.i("data", arrayAdapter.toString());
161 listView.setAdapter(arrayAdapter);
162 }
163
164 }
165

```

2.7 Layout

```
Layout.java

1 package com.example.harshit.post;
2
3 /*
4 <header>
5 Module Name      : Layout
6 Date of Creation : 10-04-2018
7 Author          : Group 2
8
9 Modification History : 11-04-2018 :- added exception for negative no. of seats
10
11 Synopsis         : This module is responsible for generating classroom layout
12
13 Global Variables  : GridView GridView;      gridview to represent seats
14                   : private int previousSelectedPosition = -1; represents previousSelectedPosition
15                   : EditText EditText;      User input for no. of seats
16                   : Button GenerateLayout    button to generate classroom layout
17
18 Functions         : void onClick    clicking the button will generate the layout based on no. of seats
19 </header>
20 */
21
22 import android.graphics.Color;
23 import android.os.AsyncTask;
24 import android.support.v7.app.AppCompatActivity;
25 import android.os.Bundle;
26 import android.util.Log;
27 import android.view.View;
28 import android.widget.AdapterView;
29 import android.widget.AdapterView.OnItemClickListener;
30 import android.widget.Button;
31 import android.widget.EditText;
32 import android.widget.GridView;
33 import android.widget.TextView;
34 import android.widget.Toast;
35
36 import java.io.BufferedReader;
37 import java.io.IOException;
38 import java.io.InputStream;
39 import java.io.OutputStream;
40 import java.io.OutputStreamWriter;
41 import java.net.HttpURLConnection;
42 import java.net.MalformedURLException;
43 import java.net.URL;
44 import java.net.URLEncoder;
45
46 public class Layout extends AppCompatActivity {
47
48     GridView GridView;
49     private int previousSelectedPosition = -1;
50     EditText EditText;
51     Button GenerateLayout;
52     @Override
53     protected void onCreate(Bundle savedInstanceState) {
54         super.onCreate(savedInstanceState); // Calling the constructor of parent class.
55         setContentView(R.layout.activity_layout); // Setting the activity content to the classroom layout view.
56         // Get the widgets reference from XML layout
57         GridView = findViewById(R.id.gridview);
58         EditText = findViewById(R.id.seat_count);
59         GenerateLayout = findViewById(R.id.generate);
60     }
61
62     public void onClick(View view){
63
64
65         String seats = EditText.getText().toString(); // extracting the string from the user input
66         // Initializing a new String Array
67         if(seats.matches(""))
68         {
69             Toast.makeText(getApplicationContext(),"Number of seats entered is empty",Toast.LENGTH_LONG).show();
70             return ;
71         }
72         int number_of_seats = Integer.parseInt(seats); // converting string to integer
73         if(number_of_seats<0)
74         {
75             Toast.makeText(getApplicationContext(),"Number of seats entered is negative",Toast.LENGTH_LONG).show();
76             return ;
77         }
78     }
79 }
```

```

73     if(number_of_seats<0)
74     {
75         Toast.makeText(getApplicationContext(),"Number of seats entered is negative",Toast.LENGTH_LONG).show();
76         return ;
77     }
78     String[] seatInfo = new String[number_of_seats];
79
80     for (int i=0;i<number_of_seats;i++)
81     {
82         seatInfo[i]=Integer.toString(i+1);
83     }
84
85     // Populate a List from Array elements
86     ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_spinner_dropdown_item, seatInfo);
87
88     GridView.setAdapter(adapter);
89     GridView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
90
91
92         @Override
93
94         public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
95             // Get the selected item text
96             String selectedItem = parent.getItemAtPosition(position).toString();
97
98             // Get the current selected view as a TextView
99             TextView tv = (TextView) view;
100
101             // Set the current selected item background color
102             tv.setBackgroundColor(Color.parseColor("#FF9A082"));
103
104             // Set the current selected item text color
105             tv.setTextColor(Color.BLUE);
106
107             // Get the last selected View from GridView
108             TextView previousSelectedView = (TextView) GridView.getChildAt(previousSelectedPosition);
109
110             // If there is a previous selected view exists
111             if (previousSelectedPosition != -1)
112             {
113                 // Set the last selected View to deselect
114                 previousSelectedView.setSelected(false);
115
116                 // Set the last selected View background color as deselected item
117                 previousSelectedView.setBackgroundColor(Color.parseColor("#FFFFFF"));
118
119                 // Set the last selected View text color as deselected item
120                 previousSelectedView.setTextColor(Color.DKGRAY);
121             }
122
123             // Set the current selected view position as previousSelectedPosition
124             previousSelectedPosition = position;
125         }
126     });
127
128     BackgroundTask backgroundTask = new BackgroundTask();
129     backgroundTask.execute(seats);
130 }
131
132 class BackgroundTask extends AsyncTask<String, Void, String> {
133
134     String ADD_SEATS_URL;
135     @Override
136     protected void onPreExecute(){
137
138         ADD_SEATS_URL = "https://22harshit.000webhostapp.com/add_seat.php";
139     }
140
141
142     @Override
143     protected String doInBackground(String... args) {
144         String seats;
145         seats = args[0];
146
147         try {
148
149             //creating a URL

```

```

140
141
142
143 @Override String doInBackground(String... args) {
144     String seats;
145     seats = args[0];
146
147     try {
148
149         //creating a URL
150         URL url = new URL(ADD_SEATS_URL);
151         //Opening the URL using HttpURLConnection
152         HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
153         // specifying the request method
154         httpURLConnection.setRequestMethod("POST");
155         httpURLConnection.setDoOutput(true);
156         // creating a output stream
157         OutputStream outputStream = httpURLConnection.getOutputStream();
158         //We will use a buffered reader to read the string from service
159         BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream, "UTF-8"));
160         // encoding the data string to be sent
161         String data_string = URLEncoder.encode("seats", "UTF-8")+"="+URLEncoder.encode(seats, "UTF-8");
162         // writing the data to the buffer
163         bufferedWriter.write(data_string);
164
165         // closing and flushing the connections
166         bufferedWriter.flush();
167         bufferedWriter.close();
168         bufferedWriter.close();
169         outputStream.close();
170         InputStream inputStream = httpURLConnection.getInputStream();
171         inputStream.close();
172         httpURLConnection.disconnect();
173
174         //finally returning the status string
175         return "Layout generated";
176
177     } catch (MalformedURLException e) {
178         e.printStackTrace();
179     } catch (IOException e) {
180         e.printStackTrace();
181     }
182     return null;
183 }
184
185 @Override
186 protected void onProgressUpdate(Void ... values){
187     super.onProgressUpdate(values);
188 }
189
190 @Override
191 protected void onPostExecute(String result)
192 {
193     // printing the success message
194     Toast.makeText(getApplicationContext(),result,Toast.LENGTH_LONG).show();
195 }
196
197 }
198
199 }
200
201

```

```

202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

3 Code Testing Team

3.1 Team profile

The code testing team comprises of the following members, all of whom are Undergraduates currently pursuing Bachelor of Technology at Indian Institute of Technology Guwahati, India in the Department of Computer Science and Engineering. All of the members are currently in the sophomore year.

1. Mitansh jain
2. Sujoy Ghosh
3. Daman Tekchandani

All members of the team are proficient in Java and have past experience in developing android applications.

4 Code Inspection Report

4.1 Code Report by Mitansh Jain

- There is use of too many global variables.
- In onClick function of Layout Activity, there is no handling of error for null value of variable number_of_seats.
- Indentation is followed properly while writing code.
- No proper bracket coding convention is used
- There are sufficient try catch statements to avoid error occurrence while running the app
- No Jump (go to) statements were used in the modules.
- All the array references used in the code were in the bound of the array.
- Proper care has been taken when there is no internet connection.
- Grid View has been chosen, it may cause problem when number of seats is high.
- The headers of each module had all the details that are required from a good header, like - Name of the module, Date on which the module was created, Author's name, Modification history, Synopsis of the module, Different functions supported, along with their input/output parameters.
- No uninitialized variables were found in any module.
- All the loops will terminate according to their terminating conditions respectively after some finite iterations.

4.2 Code Report by Sujoy Ghosh

- No index of any array is out of bound.
- JUMP (go to) statements were not found in the code.
- No proper bracket coding convention is used
- In function `GridView.setOnItemClickListener()` of activity Seat Status, no try catch statements are used for error handling. The app may crash for some Null value of seat Selection.
- All the loops used are terminating according to their terminating conditions after performing some finite iterations.
- Proper care has been taken in Indentation while writing code There are no uninitialized variables found in the module
- Grid View has been used for displaying seats, which may create problem when number of seats are high.
- In `onClick` function of Layout Activity, no care has been taken for handling NULL value of variable `number_of_seats`.
- The headers of each module have the basic details, like - Name of the module, Date on which the module was created, Author's name, Modification history, Synopsis of the module, Different functions supported, along with their input/output Parameters.
- Many global variables have been found in various modules.
- Proper care has been taken for no internet connectivity.

4.3 Code Report by Daman Tekchandani

- There were none uninitialized variables found in any module.
- All the loops terminated according to their condition. No Non-Terminating Loops were found.
- All the array references used in the code were in the bound of the array.
- No care has been taken to clear the database.
- There is no use of Jump, goto statements in any of the modules.
- While writing code, proper indentation is followed.
- Sufficient use of try catch statements to avoid error occurrence while running the app.
- No care has been taken to clear the database.
- Grid View has been used for displaying seats, which may create problem when number of seats are high.
- No proper bracket coding convention is used
- For internet connection, proper care has been taken.
- Use of too many global variables is found in various modules

5 Conclusion

The members of the code review team submitted the reports during their final meeting with the development team. From these submitted reports, we get to know about a few logical errors that were encountered during the execution of the code inspection and were listed down.

The key errors found from the code review are as follows :-

- There is a use of too many global variables which is not a good coding practice
- The problem of using GridView . For large layouts , the View may crash or behave in an unexpected manner
- There are many functions in various modules in which Exception handling is not taken care of which may lead to crashing of the application .
- No proper bracket coding convention is used .
- No care has been taken if variable (Number of seats) is having null value .