

---

# **CS-223**

## **CODE**

## **DOCUMENT**

**for**

## **Project 7**

## **Classroom Visualisation App-1**

**Prepared by:**  
**Group-12**

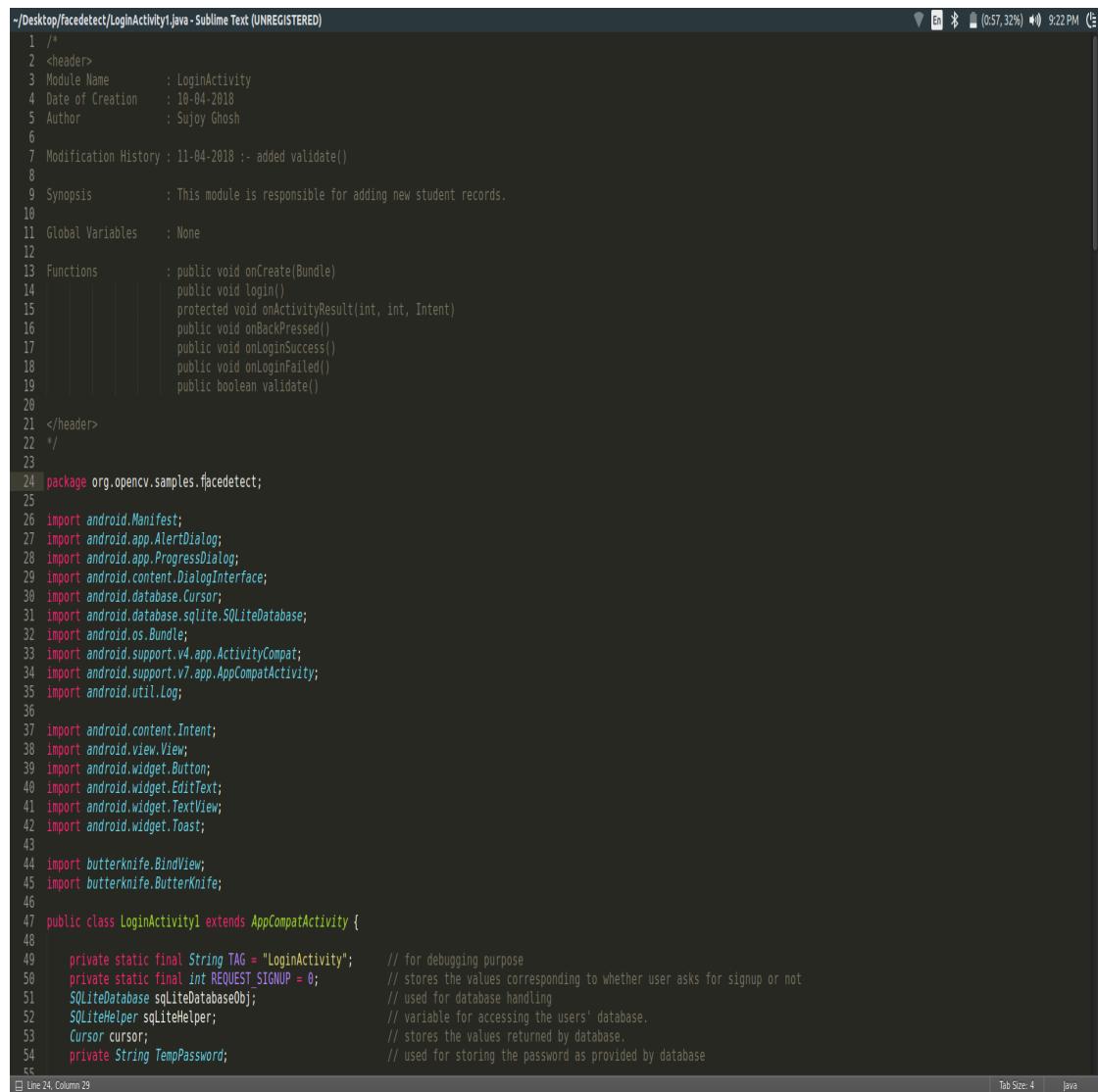
**Mitansh Jain - 160101042**  
**Sujoy Ghosh - 160101073**  
**Akul Agrawal - 160101085**

**April 22, 2018**

# **Contents**

<b>1 Login Activity</b>	<b>3</b>
<b>2 Signup Activity</b>	<b>7</b>
<b>3 Main Activity</b>	<b>12</b>
<b>4 Add Student</b>	<b>16</b>
<b>5 Add Images</b>	<b>19</b>
<b>6 Edit Student</b>	<b>22</b>
<b>7 View Student List</b>	<b>25</b>
<b>8 Delete Student</b>	<b>28</b>
<b>9 Camera Session</b>	<b>30</b>
<b>10 User Database</b>	<b>39</b>
<b>11 Student Database</b>	<b>41</b>
<b>12 Student Image Database</b>	<b>44</b>

# 1 Login Activity



The screenshot shows a Sublime Text editor window with the following details:

- File Path: ~/Desktop/facedetect/LoginActivity1.java
- Text Encoding: UTF-8
- File Type: Sublime Text (UNREGISTERED)
- System Status Bar: En (English), (0:57, 32%), 9:22 PM, battery icon

The code is a Java file for an Android application. It includes a header comment with module details, imports for various Android classes and libraries, and a class definition for LoginActivity1 extending AppCompatActivity. The class contains fields for TAG, REQUEST\_SIGNUP, sqLiteDatabaseObj, sqLiteHelper, cursor, and TempPassword, along with their respective comments.

```
1 /*
2 <header>
3 Module Name      : LoginActivity
4 Date of Creation : 10-04-2018
5 Author          : Sujoy Ghosh
6
7 Modification History : 11-04-2018 :- added validate()
8
9 Synopsis        : This module is responsible for adding new student records.
10
11 Global Variables : None
12
13 Functions       : public void onCreate(Bundle)
14           |     public void login()
15           |     protected void onActivityResult(int, int, Intent)
16           |     public void onBackPressed()
17           |     public void onLoginSuccess()
18           |     public void onLoginFailed()
19           |     public boolean validate()
20
21 </header>
22 */
23
24 package org.opencv.samples.facedetect;
25
26 import android.Manifest;
27 import android.app.AlertDialog;
28 import android.app.ProgressDialog;
29 import android.content.DialogInterface;
30 import android.database.Cursor;
31 import android.database.sqlite.SQLiteDatabase;
32 import android.os.Bundle;
33 import android.support.v4.app.ActivityCompat;
34 import android.support.v7.app.AppCompatActivity;
35 import android.util.Log;
36
37 import android.content.Intent;
38 import android.view.View;
39 import android.widget.Button;
40 import android.widget.EditText;
41 import android.widget.TextView;
42 import android.widget.Toast;
43
44 import butterknife.BindView;
45 import butterknife.ButterKnife;
46
47 public class LoginActivity1 extends AppCompatActivity {
48
49     private static final String TAG = "LoginActivity";           // for debugging purpose
50     private static final int REQUEST_SIGNUP = 0;                // stores the values corresponding to whether user asks for signup or not
51     SQLiteDatabase sqLiteDatabaseObj;                          // used for database handling
52     SQLiteHelper sqLiteHelper;                                // variable for accessing the users' database.
53     Cursor cursor;                                         // stores the values returned by database.
54     private String TempPassword;                            // used for storing the password as provided by database
55 }
```

```

-/Desktop/facetedetect/LoginActivity1.java - Sublime Text (UNREGISTERED)
52     SQLiteHelper sqliteHelper;           // variable for accessing the users' database.
53     Cursor cursor;                     // stores the values returned by database.
54     private String TempPassword;        // used for storing the password as provided by database
55
56     @BindView(R.id.input_email) EditText editText_emailText; // binding the textbox with name _emailText
57     @BindView(R.id.input_password) EditText editText_passwordText; // binding the textbox with name _passwordText
58     @BindView(R.id.btn_login) Button loginButton; // binding the button with name loginButton
59     @BindView(R.id.link_signup) TextView signupLink; // binding the label with name _signupLink
60
61     @Override
62     public void onCreate(Bundle savedInstanceState) {           // this functions overrides the function defined in the parent class "AppCompatActivity".
63         super.onCreate(savedInstanceState);                      // this method is called when the activity is first created.
64         setContentView(R.layout.activity_login);               // creates the activity.
65         int PERMIT_ALL = 1;                                  // sets the content UI to be displayed on screen.
66         // give permissions to all required permissions.
67         // contains all permissions
68         String[] PERMISSIONS = {Manifest.permission.CAMERA, Manifest.permission.WRITE_EXTERNAL_STORAGE};
69         ActivityCompat.requestPermissions(this, PERMISSIONS, PERMIT_ALL); // asks for granting permissions if not done already.
70         Butterknife.bind(this);                            // handle all ui elements.
71         sqliteHelper = new SQLiteHelper(this);            // creates instances of users' database for accessing.
72         loginButton.setOnClickListener(new View.OnClickListener() { // this creates the constructor for onClick().
73             @Override                                         // this functions overrides the function defined in the parent class "AppCompatActivity".
74             public void onClick(View v) {                   // this function defines the tasks to be done when button is clicked.
75                 login();                                // run the login method on clicking login button
76             }
77         });
78         signupLink.setOnClickListener(new View.OnClickListener() { // this creates the constructor for onClick().
79             @Override                                         // this functions overrides the function defined in the parent class "AppCompatActivity".
80             public void onClick(View v) {                   // this function defines the tasks to be done when button is clicked.
81                 Intent intent = new Intent(getApplicationContext(), SignupActivity.class); // Start the Signup activity
82                 startActivityForResult(intent, REQUEST_SIGNUP); // start the signup activity.
83                 finish();                                // finish the present activity, clearing memory usage and decreasing ram usage
84                 overridePendingTransition(R.anim.push_left_in, R.anim.push_left_out); // defines the animation of moving from one activity to another.
85             }
86         });
87     }
88 }
89
90     public void login() {                                // this method handles login of the user
91         // Log.d(TAG, "Login");
92         // if all fields are not valid entries, then display login failed message.
93         if (!validate()) {
94             onLoginFailed();                           // display the Login failed message.
95             return;
96         }
97
98         _loginButton.setEnabled(false);              // if the form is valid, then disable the login button.
99         // start the progressbar
100        final ProgressDialog progressDialog = new ProgressDialog(LoginActivity1.this,
101                    R.style.AppTheme_Dark_Dialog);
102        progressDialog.setIndeterminate(true);       // keep it running till the process completes
103        progressDialog.setMessage("Authenticating..."); // set the message to be displayed
104        progressDialog.show();                      // display the progress bar
105
106

```

Line 24, Column 29

Tab Size: 4 Java

```

~/Desktop/facetedetect/LoginActivity.java - Sublime Text (UNREGISTERED)          // (0:55,31%) 9:22PM (5)
100    final ProgressDialog progressDialog = new ProgressDialog(LoginActivity.this,           // start the progressbar
101        R.style.AppTheme_Dark_Dialog);
102    progressDialog.setIndeterminate(true);           // keep it running till the process completes
103    progressDialog.setMessage("Authenticating...");
104    progressDialog.show();                         // display the progress bar
105
106    String email = _emailText.getText().toString(); // obtain the email as entered by user
107    final String password = _passwordText.getText().toString(); // fetch the password as entered by user
108
109    Cursor result = sqliteHelper.getPassword(email); // get the password corresponding to given email id.
110    if(result.getCount() == 0) {                     // if there are no records with given email id
111        new android.os.Handler().postDelayed(      // this defines what to be performed under this circumstance
112            new Runnable() {                         // define what to run
113                public void run() {
114                    onLoginFailed();                  // display that login has failed.
115                    progressDialog.dismiss();       // close the progress bar
116                }
117            }, 3000);
118        return;
119    }
120    result.moveToFirst();                          // else move to first row
121    TempPassword = result.getString(3);          // extract the password from it.
122
123    new android.os.Handler().postDelayed(         // this defines what to be performed under this circumstance
124        new Runnable() {                         // defines what to run on login success/failed
125            public void run() {
126                if(TempPassword.equals(password)) { // if extracted password equals entered password, call onLoginSuccess()
127                    onLoginSuccess();
128                } else {                      // if not, then call onLoginFailed()
129                    onLoginFailed();
130                }
131                progressDialog.dismiss();       // close the progress dialog
132            }
133        }, 3000);
134    }
135}
136
137
138 @Override                                     // this functions overrides the function defined in the parent class "AppCompatActivity".
139 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
140     if (requestCode == REQUEST_SIGNUP) {           // if signup is successful then finish the login activity and progress to main activity
141         if (resultCode == RESULT_OK) {
142             this.finish();
143         }
144     }
145 }
146
147 @Override                                     // this functions overrides the function defined in the parent class "AppCompatActivity".
148 public void onBackPressed() {
149     new AlertDialog.Builder(this)               // sets the icon for dialog box
150         .setIcon(android.R.drawable.ic_dialog_alert) // sets the title for the dialog box
151         .setTitle("Close App")
152         .setMessage("Are you sure you want to exit the app?") // defines tasks to be performed when user clicks yes.
153         .setPositiveButton("Yes", new DialogInterface.OnClickListener()
154             {

```

Tab Size: 4 Java

```

-/Desktop/facetedetct/LoginActivity.java - Sublime Text (UNREGISTERED)
148     public void onBackPressed() {
149         new AlertDialog.Builder(this) // sets the icon for dialog box
150             .setIcon(android.R.drawable.ic_dialog_alert) // sets the title for the dialog box
151             .setTitle("Close App")
152             .setMessage("Are you sure you want to exit the app?") // defines tasks to be performed when user clicks yes.
153             .setPositiveButton("Yes", new DialogInterface.OnClickListener()
154             {
155                 @Override
156                 public void onClick(DialogInterface dialog, int which) {
157                     moveTaskToBack(true); // prevent going back to main menu
158                 }
159             })
160             .setNegativeButton("No", null) // if the user selects no, then do nothing
161         .show();
162     }
163 }
164
165     public void onLoginSuccess() { // this method handles tasks to be performed on successful login
166         _loginButton.setEnabled(true); // enable the login button
167         Intent intent = new Intent(getApplicationContext(), MainActivity.class); // define the next activity to start
168         startActivity(intent); // start the new activity
169         finish(); // finish this activity
170     }
171 }
172
173     public void onLoginFailed() { // this method handles tasks to be performed on failed login
174         Toast.makeText(getApplicationContext(), "Login failed", Toast.LENGTH_LONG).show(); // display the message of failed login
175         _loginButton.setEnabled(true); // enable the login button
176     }
177 }
178
179     public boolean validate() { // this method handles the form validation
180         boolean valid = true; // initialise the valid variable, assuming the form is valid, set to true.
181
182         String email = _emailText.getText().toString(); // fetch the email from the textbox
183         String password = _passwordText.getText().toString(); // fetch the password from the textbox
184         // checks whether it follows proper email syntax and is not empty
185         if (email.isEmpty() || android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
186             _emailText.setError("Enter a valid email address"); // if it doesn't follow, then display the error
187             valid = false; // set valid to false
188         } else { // if it follows proper guidelines, then don't display any error
189             _emailText.setError(null);
190         }
191         // checks whether password is not empty and has length between 4 to 10 characters
192         if (password.isEmpty() || password.length() < 4 || password.length() > 10) {
193             _passwordText.setError("Between 4 and 10 alphanumeric characters");
194             valid = false; // if it doesn't follow, then display the error, and make valid equal to false
195         } else {
196             _passwordText.setError(null); // if it follows proper guidelines, then don't display any error
197         }
198
199         return valid; // return the value denoting validity of form
200     }
201 }
202

```

Line 24, Column 29 | Tab Size: 4 | Java

## 2 Signup Activity

```

-/Desktop/facetedet/SigupActivity.java -- Sublime Text (UNREGISTERED)
52     private static final String TAG = "SignupActivity";
53
54     @BindView(R.id.input_name) EditText _nameText;
55     @BindView(R.id.input_email) EditText _emailText;
56     @BindView(R.id.input_password) EditText _passwordText;
57     @BindView(R.id.input_reEnterPassword) EditText _reEnterPasswordText;
58     @BindView(R.id.btn_signup) Button _signupButton;
59     @BindView(R.id.link_login) TextView _loginLink;
60
61     private Boolean EditTextEmptyHolder;
62     private SQLiteDatabase sqliteDatabaseObj;
63     private SQLiteHelper sqliteHelper;
64     private Cursor cursor;
65     private String F_Result = "Not_Found";
66     private String name, email, password, reEnterPassword;
67
68
69     @Override
70     public void onCreate(Bundle savedInstanceState) {
71         super.onCreate(savedInstanceState);
72         setContentView(R.layout.activity_signup);
73         ButterKnife.bind(this);
74         sqliteHelper = new SQLiteHelper(this);
75         _signupButton.setOnClickListener(new View.OnClickListener() {
76             @Override
77             public void onClick(View v) {
78                 signup();
79             }
80         });
81
82         _loginLink.setOnClickListener(new View.OnClickListener() {
83             @Override
84             public void onClick(View v) {
85
86                 Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
87                 startActivity(intent);
88                 finish();
89                 // Finish the registration screen and return to the login activity.
90                 overridePendingTransition(R.anim.push_left_in, R.anim.push_left_out);
91             }
92         });
93     }
94
95     public void signup() {
96         EditTextEmptyHolder = false;
97         if (!validate()) {
98             onSignupFailed();
99             return;
100        }
101        EditTextEmptyHolder = true;
102        _signupButton.setEnabled(false);
103
104        final ProgressDialog progressDialog = new ProgressDialog(SignupActivity.this,
105                    R.style.AppTheme_Dark_Dialog);
106        progressDialog.setIndeterminate(true);
107
108        progressDialog.show();
109
110        _nameText.setText("");
111        _emailText.setText("");
112        _passwordText.setText("");
113        _reEnterPasswordText.setText("");
114
115        _nameText.setError(null);
116        _emailText.setError(null);
117        _passwordText.setError(null);
118        _reEnterPasswordText.setError(null);
119
120        _nameText.requestFocus();
121
122        _passwordText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
123            @Override
124            public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
125                if (actionId == EditorInfo.IME_ACTION_DONE) {
126                    _reEnterPasswordText.requestFocus();
127                }
128                return false;
129            }
130        });
131
132        _reEnterPasswordText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
133            @Override
134            public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
135                if (actionId == EditorInfo.IME_ACTION_DONE) {
136                    _signupButton.setEnabled(true);
137                }
138                return false;
139            }
140        });
141
142        _signupButton.setOnClickListener(new View.OnClickListener() {
143            @Override
144            public void onClick(View v) {
145                if (_nameText.getText().toString().trim().length() > 0 & _emailText.getText().toString().trim().length() > 0 & _passwordText.getText().toString().trim().length() > 0 & _reEnterPasswordText.getText().toString().trim().length() > 0) {
146
147                    if (_passwordText.getText().toString().trim().equals(_reEnterPasswordText.getText().toString().trim())) {
148
149                        if (EditTextEmptyHolder) {
150                            SQLiteDatabase databaseObj = sqliteHelper.getWritableDatabase();
151
152                            ContentValues contentValues = new ContentValues();
153
154                            contentValues.put("name", name);
155                            contentValues.put("email", email);
156                            contentValues.put("password", password);
157
158                            databaseObj.insert("users", null, contentValues);
159
160                            F_Result = "Success";
161
162                            progressDialog.dismiss();
163
164                            Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
165                            startActivity(intent);
166                            finish();
167
168                            _nameText.setText("");
169                            _emailText.setText("");
170                            _passwordText.setText("");
171                            _reEnterPasswordText.setText("");
172
173                            _nameText.setError(null);
174                            _emailText.setError(null);
175                            _passwordText.setError(null);
176                            _reEnterPasswordText.setError(null);
177
178                            _nameText.requestFocus();
179
180                            _passwordText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
181                                @Override
182                                public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
183                                    if (actionId == EditorInfo.IME_ACTION_DONE) {
184                                        _reEnterPasswordText.requestFocus();
185                                    }
186                                    return false;
187                                }
188                            });
189
190                            _reEnterPasswordText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
191                                @Override
192                                public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
193                                    if (actionId == EditorInfo.IME_ACTION_DONE) {
194                                        _signupButton.setEnabled(true);
195                                    }
196                                    return false;
197                                }
198                            });
199
200                            _signupButton.setOnClickListener(new View.OnClickListener() {
201                                @Override
202                                public void onClick(View v) {
203
204                                if (_nameText.getText().toString().trim().length() > 0 & _emailText.getText().toString().trim().length() > 0 & _passwordText.getText().toString().trim().length() > 0 & _reEnterPasswordText.getText().toString().trim().length() > 0) {
205
206                                    if (_passwordText.getText().toString().trim().equals(_reEnterPasswordText.getText().toString().trim())) {
207
208                                        if (EditTextEmptyHolder) {
209                                            SQLiteDatabase databaseObj = sqliteHelper.getWritableDatabase();
210
211                                            ContentValues contentValues = new ContentValues();
212
213                                            contentValues.put("name", name);
214                                            contentValues.put("email", email);
215                                            contentValues.put("password", password);
216
217                                            databaseObj.insert("users", null, contentValues);
218
219                                            F_Result = "Success";
220
221                                            progressDialog.dismiss();
222
223                                            Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
224                                            startActivity(intent);
225                                            finish();
226
227                                            _nameText.setText("");
228                                            _emailText.setText("");
229                                            _passwordText.setText("");
230                                            _reEnterPasswordText.setText("");
231
232                                            _nameText.setError(null);
233                                            _emailText.setError(null);
234                                            _passwordText.setError(null);
235                                            _reEnterPasswordText.setError(null);
236
237                                            _nameText.requestFocus();
238
239                                            _passwordText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
240                                                @Override
241                                                public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
242                                                    if (actionId == EditorInfo.IME_ACTION_DONE) {
243                                                        _reEnterPasswordText.requestFocus();
244                                                    }
245                                                    return false;
246                                                }
247                                            });
248
249                                            _reEnterPasswordText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
250                                                @Override
251                                                public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
252                                                    if (actionId == EditorInfo.IME_ACTION_DONE) {
253                                                        _signupButton.setEnabled(true);
254                                                    }
255                                                    return false;
256                                                }
257                                            });
258
259                                            _signupButton.setOnClickListener(new View.OnClickListener() {
260                                                @Override
261                                                public void onClick(View v) {
262
263                                                    if (_nameText.getText().toString().trim().length() > 0 & _emailText.getText().toString().trim().length() > 0 & _passwordText.getText().toString().trim().length() > 0 & _reEnterPasswordText.getText().toString().trim().length() > 0) {
264
265                                                        if (_passwordText.getText().toString().trim().equals(_reEnterPasswordText.getText().toString().trim())) {
266
267                                                            if (EditTextEmptyHolder) {
268                                                                SQLiteDatabase databaseObj = sqliteHelper.getWritableDatabase();
269
270                                                                ContentValues contentValues = new ContentValues();
271
272                                                                contentValues.put("name", name);
273                                                                contentValues.put("email", email);
274                                                                contentValues.put("password", password);
275
276                                                                databaseObj.insert("users", null, contentValues);
277
278                                                                F_Result = "Success";
279
280                                                                progressDialog.dismiss();
281
282                                                                Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
283                                                                startActivity(intent);
284                                                                finish();
285
286                                                                _nameText.setText("");
287                                                                _emailText.setText("");
288                                                                _passwordText.setText("");
289                                                                _reEnterPasswordText.setText("");
290
291                                                                _nameText.setError(null);
292                                                                _emailText.setError(null);
293                                                                _passwordText.setError(null);
294                                                                _reEnterPasswordText.setError(null);
295
296                                                                _nameText.requestFocus();
297
298                                                                _passwordText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
299                                                                    @Override
300                                                                    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
301                                                                        if (actionId == EditorInfo.IME_ACTION_DONE) {
302                                                                            _reEnterPasswordText.requestFocus();
303                                                                        }
304                                                                        return false;
305                                                                    }
306                                                                });
307
308                                                                _reEnterPasswordText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
309                                                                    @Override
310                                                                    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
311                                                                        if (actionId == EditorInfo.IME_ACTION_DONE) {
312                                                                            _signupButton.setEnabled(true);
313                                                                        }
314                                                                        return false;
315                                                                    }
316                                                                });
317
318                                                                _signupButton.setOnClickListener(new View.OnClickListener() {
319                                                                    @Override
320                                                                    public void onClick(View v) {
321
322                                                                        if (_nameText.getText().toString().trim().length() > 0 & _emailText.getText().toString().trim().length() > 0 & _passwordText.getText().toString().trim().length() > 0 & _reEnterPasswordText.getText().toString().trim().length() > 0) {
323
324                                                                            if (_passwordText.getText().toString().trim().equals(_reEnterPasswordText.getText().toString().trim())) {
325
326                                                                                if (EditTextEmptyHolder) {
327                                                                                    SQLiteDatabase databaseObj = sqliteHelper.getWritableDatabase();
328
329                                                                                    ContentValues contentValues = new ContentValues();
330
331                                                                                    contentValues.put("name", name);
332                                                                                    contentValues.put("email", email);
333                                                                                    contentValues.put("password", password);
334
335                                                                                    databaseObj.insert("users", null, contentValues);
336
337                                                                                    F_Result = "Success";
338
339                                                                                    progressDialog.dismiss();
340
341                                                                                    Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
342                                                                                    startActivity(intent);
343                                                                                    finish();
344
345                                                                                    _nameText.setText("");
346                                                                                    _emailText.setText("");
347                                                                                    _passwordText.setText("");
348                                                                                    _reEnterPasswordText.setText("");
349
350                                                                                    _nameText.setError(null);
351                                                                                    _emailText.setError(null);
352                                                                                    _passwordText.setError(null);
353                                                                                    _reEnterPasswordText.setError(null);
354
355                                                                                    _nameText.requestFocus();
356
357                                                                                    _passwordText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
358                                                                                        @Override
359                                                                                        public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
360                                                                                            if (actionId == EditorInfo.IME_ACTION_DONE) {
361                                                                                                _reEnterPasswordText.requestFocus();
362                                                                                            }
363                                                                                            return false;
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
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
```

```

~/Desktop/facetedetect/SignupActivity.java -- Sublime Text (UNREGISTERED)
100    }
101    EditTextEmptyHolder = true;
102    signupButton.setEnabled(false);
103    // start the progressbar
104    final ProgressDialog progressDialog = new ProgressDialog(SignupActivity.this,
105        R.style.AppTheme_Dark_Dialog);
106    progressDialog.setIndeterminate(true);
107    progressDialog.setMessage("Creating Account...");
108    progressDialog.show();
109
110    name = _nameText.getText().toString();
111    email = _emailText.getText().toString();
112    password = _passwordText.getText().toString();
113    reEnterPassword = _reEnterPasswordText.getText().toString();
114
115    SQLiteDataBaseBuild();
116    SQLiteTableBuild();
117    EmptyEditTextAfterDataInsert();
118
119    new android.os.Handler().postDelayed(
120        new Runnable() {
121            public void run() {
122                CheckingEmailAlreadyExistsOrNot();
123                progressDialog.dismiss();
124            }
125        }, 3000);
126    }
127
128
129
130    public void onSignupSuccess() {
131        _signupButton.setEnabled(true);
132        setResult(RESULT_OK, null);
133        Intent intent = new Intent(getApplicationContext(), MainActivity.class);
134        startActivity(intent);
135        finish();
136    }
137
138    public void onSignupFailed() {
139        Toast.makeText(getApplicationContext(), "signup failed", Toast.LENGTH_LONG).show(); // display the message of failed signup
140        _signupButton.setEnabled(true); // enable the signup button
141    }
142
143    @Override
144    public void onBackPressed() {
145        new AlertDialog.Builder(this)
146            .setIcon(android.R.drawable.ic_dialog_alert) // launches new alert dialog box
147            .setTitle("Close App") // sets the icon for dialog box
148            .setMessage("Are you sure you want to exit the app?") // sets the title for the dialog box
149            .setPositiveButton("Yes", new DialogInterface.OnClickListener()
150            {
151                @Override
152                public void onClick(DialogInterface dialog, int which) {
153                    moveTaskToBack(true); // prevent going back to main menu
154                }
155            });
156    }

```

Line 133, Column 97

Tab Size: 4 Java

```

~/Desktop/facel detect/SigupActivity.java -- Sublime Text (UNREGISTERED)
151     @Override
152         public void onClick(DialogInterface dialog, int which) {
153             moveTaskToBack(true);                                // prevent going back to main menu
154         }
155     })
156     .setNegativeButton("No", null)                         // if the user selects no, then do nothing
157     .show();
158 }
159 }
160
161 // SQLITE database build method.
162 public void SQLiteDataBaseBuild(){
163     SQLiteDatabaseObj = openOrCreateDatabase(SQLiteHelper.DATABASE_NAME, Context.MODE_PRIVATE, null);           // creates the database if it doesn't exist already
164 }
165
166 // SQLITE table build method.
167 public void SQLiteTableBuild() {
168     SQLiteDatabaseObj.execSQL("CREATE TABLE IF NOT EXISTS " + SQLiteHelper.TABLE_NAME + "(" + SQLiteHelper.Table.Column.ID +
169     " INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, " + SQLiteHelper.Table.Column_1_Name + " VARCHAR, " + SQLiteHelper.Table.Column_2_Email +
170     " VARCHAR, " + SQLiteHelper.Table.Column_3_Password + " VARCHAR);");                                         // creates the table if it doesn't exist
171 }
172
173
174
175 // Empty edittext after done inserting process method.
176 public void EmptyEditTextAfterDataInsert(){
177     _nameText.getText().clear();                           // clear the name textbox
178     _emailText.getText().clear();                         // clear the email textbox
179     _passwordText.getText().clear();                     // clear the password textbox
180     _reEnterPasswordText.getText().clear();              // clear the confirm-password textbox
181 }
182
183
184 public void CheckingEmailAlreadyExistsOrNot(){          // This method is responsible for checking if Email already exists or not.
185     boolean ifEmailExists = SQLiteHelper.ifEmailExists(email);                                     // Adding search email query to cursor.
186     if(ifEmailExists){                           // If Email is already exists then Result variable value set as Email Found.
187         F_Result = "Email Found";
188     }
189
190     CheckFinalResult();                           // Calling method to check final result and insert data into SQLite database.
191 }
192
193 }
194
195 public void CheckFinalResult(){                        // this method is responsible for check final result.
196
197     if(F_Result.equalsIgnoreCase("Email Found")) {      // Checking whether email is already exists or not.
198         // If email is exists then dispaly the corresponding message.
199         Toast.makeText(SigupActivity.this,"Email Already Exists",Toast.LENGTH_LONG).show();
200         onSignupFailed();                             // call signupFailed().
201     } else {                                       // if the corresponding details doesn't exist in database.
202         boolean val = SQLiteHelper.insertData(name, email, password);           // user registration details will entered to SQLite database.
203         if(val) {                                 // if successfully inserted.
204             onSignupSuccess();                      // call signupSuccess().
205         } else {                                // if insertion not successful, then call signupFailed()
206     }

```

Line 133, Column 97 Tab Size: 4 Java

```

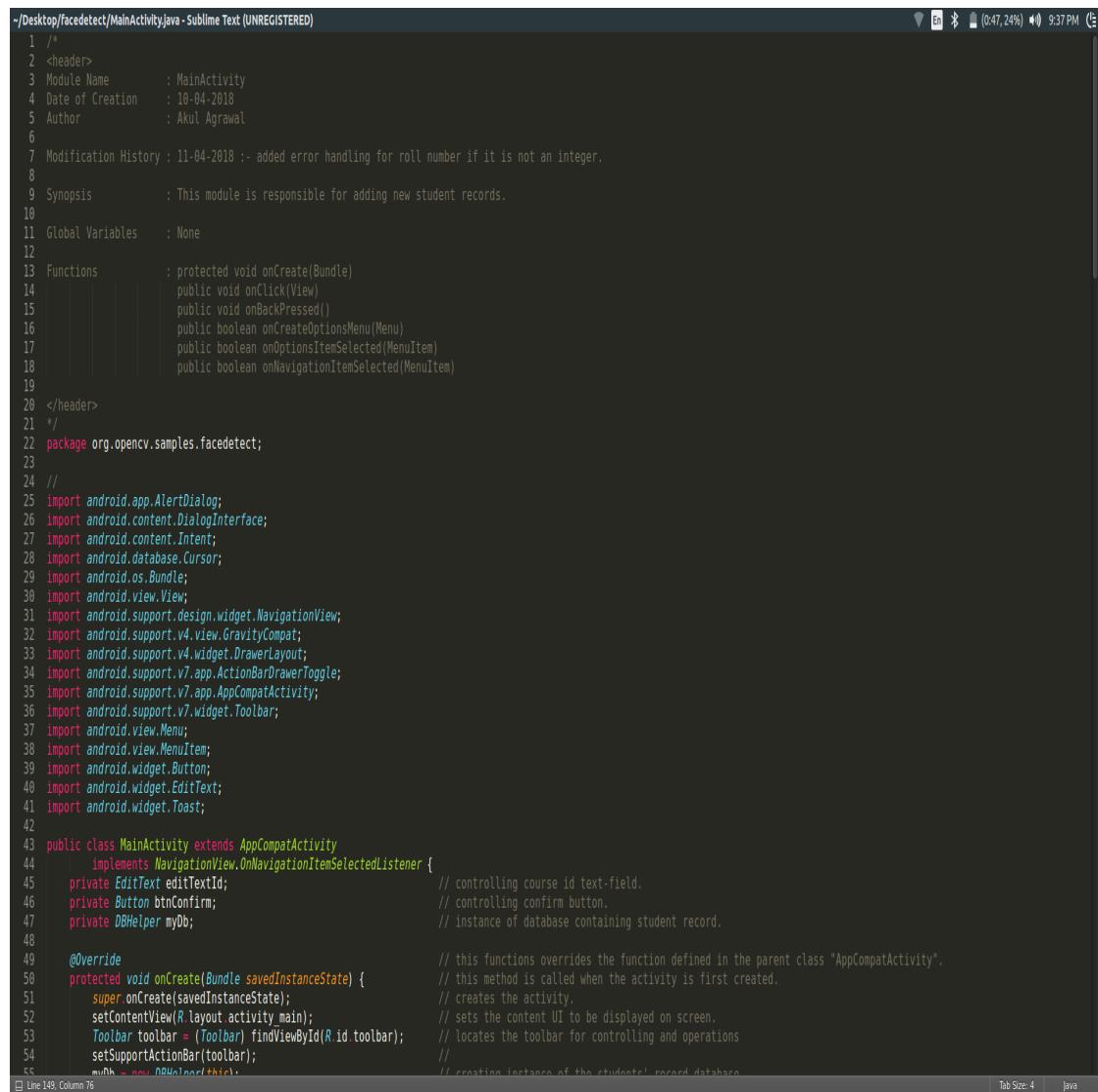
~/Desktop/facetedetect/SignupActivity.java -- Sublime Text (UNREGISTERED)
199     Toast.makeText(SignupActivity.this,"Email Already Exists",Toast.LENGTH_LONG).show();
200     onSignupFailed(); // call signupFailed().
201 } else { // if the corresponding details doesn't exist in database.
202     boolean val = sqliteHelper.insertData(name, email, password); // user registration details will entered to SQLite database.
203     if(val) { onSignupSuccess(); // if successfully inserted. }
204     } else { onSignupFailed(); // if insertion not successful, then call signupFailed() }
205 }
206 }
207 }
208 }
209 F_Result = "Not_Found" ;
210 }
211
212
213
214 public boolean validate() {
215     boolean valid = true; // this method is responsible for form data validation
216     String name = _nameText.getText().toString(); // initialise the valid variable, assuming the form is valid, set to true.
217     String email = _emailText.getText().toString(); // fetch the name from the textbox
218     String password = _passwordText.getText().toString(); // fetch the email from the textbox
219     String reEnterPassword = _reEnterPasswordText.getText().toString(); // fetch the password from the textbox
220
221     if (name.isEmpty() || name.length() < 3) { // fetch the confirm-password from the textbox
222         _nameText.setError("at least 3 characters"); // checks whether name is empty or length is less than 3 letters.
223         valid = false; // set the error message to be displayed
224     } else { // set valid to false
225         _nameText.setError(null); // if the entered is correct as per guidelines, then no error is displayed
226     }
227
228     if (email.isEmpty() || !android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) { // checks whether it follows proper email syntax and is not empty
229         _emailText.setError("enter a valid email address"); // if it doesnt follow, then display the error
230         valid = false; // set valid to false
231     } else { // if it follows proper guidelines, then don't display any error
232         _emailText.setError(null);
233     }
234
235     if (password.isEmpty() || password.length() < 4 || password.length() > 10) { // checks whether password is not empty and has length between 4 to 10 characters
236         _passwordText.setError("between 4 and 10 alphanumeric characters"); // if it doesnt follow, then display the error, and make valid equal to false
237         valid = false;
238     } else { // if it follows proper guidelines, then don't display any error
239         _passwordText.setError(null);
240     }
241
242     if (reEnterPassword.isEmpty() || reEnterPassword.length() < 4 || reEnterPassword.length() > 10 || !(reEnterPassword.equals(password))) { // checks whether password is not empty and has length between 4 to 10 characters
243         _reEnterPasswordText.setError("Password Do not match"); // and equals the password entered
244         valid = false; // if it doesn't follow, then display the error, and make valid equal to false
245     } else { // if it follows proper guidelines, then don't display any error
246         _reEnterPasswordText.setError(null);
247     }
248
249     return valid; // return the value denoting validity of form
250 }
251 }
252 }

```

Line 133, Column 97

Tab Size: 4 Java

### 3 Main Activity



The screenshot shows a Sublime Text editor window with the file `MainActivity.java` open. The code is a Java class for an Android application. It includes comments at the top describing the module's purpose, creation date, author, and modification history. The class extends `AppCompatActivity` and implements `NavigationView.OnNavigationItemSelectedListener`. It contains imports for various Android components like `AlertDialog`, `DialogInterface`, `Intent`, `Cursor`, `Bundle`, `View`, `NavigationView`, `GravityCompat`, `DrawerLayout`, `ActionBarDrawerToggle`, `AppCompatActivity`, `Toolbar`, `Menu`, `MenuItem`, `Button`, `EditText`, and `Toast`. The class has private fields for `EditText`, `Button`, and `DBHelper`. The `onCreate` method initializes the activity, sets the content view, and creates a toolbar. The `onNavigationItemSelected` method is overridden to handle item selection.

```
1 /*
2 <header>
3 Module Name      : MainActivity
4 Date of Creation : 10-04-2018
5 Author          : Akul Agrawal
6
7 Modification History : 11-04-2018 :- added error handling for roll number if it is not an integer.
8
9 Synopsis        : This module is responsible for adding new student records.
10
11 Global Variables : None
12
13 Functions       : protected void onCreate(Bundle)
14           public void onClick(View)
15           public void onBackPressed()
16           public boolean onCreateOptionsMenu(Menu)
17           public boolean onOptionsItemSelected(MenuItem)
18           public boolean onNavigationItemSelected(MenuItem)
19
20 </header>
21 */
22 package org.opencv.samples.facedetect;
23
24 //
25 import android.app.AlertDialog;
26 import android.content.DialogInterface;
27 import android.content.Intent;
28 import android.database.Cursor;
29 import android.os.Bundle;
30 import android.view.View;
31 import android.support.design.widget.NavigationView;
32 import android.support.v4.view.GravityCompat;
33 import android.support.v4.widget.DrawerLayout;
34 import android.support.v7.app.ActionBarDrawerToggle;
35 import android.support.v7.app.AppCompatActivity;
36 import android.support.v7.widget.Toolbar;
37 import android.view.Menu;
38 import android.view.MenuItem;
39 import android.widget.Button;
40 import android.widget.EditText;
41 import android.widget.Toast;
42
43 public class MainActivity extends AppCompatActivity
44     implements NavigationView.OnNavigationItemSelected {
45     private EditText editTxtId; // controlling course id text-field.
46     private Button btnConfirm; // controlling confirm button.
47     private DBHelper myDb; // instance of database containing student record.
48
49     @Override // this functions overrides the function defined in the parent class "AppCompatActivity".
50     protected void onCreate(Bundle savedInstanceState) { // this method is called when the activity is first created.
51         super.onCreate(savedInstanceState); // creates the activity.
52         setContentView(R.layout.activity_main); // sets the content UI to be displayed on screen.
53         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar); // locates the toolbar for controlling and operations
54         setSupportActionBar(toolbar); // creating instance of the student's record database
55         myDb = new DBHelper(this);
56
57     }
58
59     @Override // this method is called when an item in the navigation menu is selected.
60     public void onNavigationItemSelected(@NonNull MenuItem item) {
61
62     }
63
64 }
```

```

-/Desktop/facdetecf/MainActivity.java - Sublime Text (UNREGISTERED)
40 import android.widget.EditText;
41 import android.widget.Toast;
42
43 public class MainActivity extends AppCompatActivity
44     implements NavigationView.OnNavigationItemSelected {
45     private EditText editTextId; // controlling course id text-field.
46     private Button btnConfirm; // controlling confirm button.
47     private DBHelper myDb; // instance of database containing student record.
48
49     @Override // this functions overrides the function defined in the parent class "AppCompatActivity".
50     protected void onCreate(Bundle savedInstanceState) { // this method is called when the activity is first created.
51         super.onCreate(savedInstanceState); // creates the activity.
52         setContentView(R.layout.activity_main); // sets the content UI to be displayed on screen.
53         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar); // locates the toolbar for controlling and operations
54         setSupportActionBar(toolbar); // creating instance of the students' record database.
55         myDb = new DBHelper(this); // new DBHelper(this);
56         DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout); // locates the navigation drawer for controlling
57                                         // for toggling between opening and closing of navigation drawer
58         ActionBarDrawerToggle toggle = new ActionBarDrawerToggle( // this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
59         drawer.addDrawerListener(toggle); // the toggle state is adder as option into drawer
60         toggle.syncState(); // syncing it with drawer
61         NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view); // call the drawer
62         navigationView.setNavigationItemSelectedListener(this); // attach drawer with this main activity
63         editTextId = (EditText) findViewById(R.id.courseid); // locates the textfield for fetching purpose
64         btnConfirm = (Button) findViewById(R.id.sessionstart); // locates the button for controlling purpose
65
66         btnConfirm.setOnClickListener( // new process starts if the button is clicked.
67             new View.OnClickListener() // this creates the constructor for onClick().
68                 @Override // this functions overrides the function defined in the parent class "AppCompatActivity".
69                 public void onClick(View v) { // this function defines the tasks to be done when button is clicked.
70                     String courseId = editTextId.getText().toString(); // extracts the course id from the textfield
71                     Cursor result = myDb.getCourseData(courseId); // if (courseID.isEmpty()) {
72
73                     if (courseID.isEmpty()) { // Toast.makeText(MainActivity.this, "Field(s) is empty", Toast.LENGTH_LONG).show();
74                         } else if(result.getCount() > 0) { // check whether if there are students with given course id
75                             // if there is, define new activity to start next
76                             Intent intent = new Intent(MainActivity.this, FdActivity.class); // put the course id as parameter to it
77                             intent.putExtra("id", courseId); // start the next activity
78                             startActivity(intent); // if there are no students with given course id, then print the message
79                             } else { // Toast.makeText(getApplicationContext(), "No such courseID exists", Toast.LENGTH_LONG).show();
80                             }
81                         }
82                     }
83                 }
84             );
85         );
86     }
87 }
88
89     @Override // this functions overrides the function defined in the parent class "AppCompatActivity".
90     public void onBackPressed() { // this method handles the process when user press back button
91         DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout); // locates the navigation drawer for controlling
92         if (drawer.isDrawerOpen(GravityCompat.START)) { // if drawer is open
93             drawer.closeDrawer(GravityCompat.START); // then close the drawer
94         }

```

Tab Size: 4 Java

```

~/Desktop/facetect/MainActivity.java - Sublime Text (UNREGISTERED)
  85     }
  86   );
  87 }
  88
  89 @Override
 90 public void onBackPressed() {
 91   DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
 92   if (drawer.isDrawerOpen(GravityCompat.START)) {
 93     drawer.closeDrawer(GravityCompat.START);
 94   } else {
 95     new AlertDialog.Builder(this)
 96       .setIcon(android.R.drawable.ic_dialog_alert)
 97       .setTitle("Log-out")
 98       .setMessage("Are you sure you want to logout?")
 99       .setPositiveButton("Yes", new DialogInterface.OnClickListener()
100       {
101         @Override
102         public void onClick(DialogInterface dialog, int which) { // specifies the task to be performed when yes is clicked
103           Intent intent = new Intent(MainActivity.this, LoginActivity.class);
104           startActivity(intent); // define the next activity and go to next activity.
105           finish(); // finish this activity
106         }
107       })
108     .setNegativeButton("No", null)
109     .show();
110   }
111 }
112
113
114 @Override
115 public boolean onCreateOptionsMenu(Menu menu) {
116   getMenuInflater().inflate(R.menu.main, menu);
117   return true;
118 }
119
120
121 @Override
122 public boolean onOptionsItemSelected(MenuItem item) {
123   // this functions overrides the function defined in the parent class "Menu".
124   // Handle action bar item clicks here. The action bar will
125   // automatically handle clicks on the Home/Up button, so long
126   // as you specify a parent activity in AndroidManifest.xml.
127   int id = item.getItemId();
128   if (id == R.id.action_settings) {
129     return true;
130   }
131   return super.onOptionsItemSelected(item); // returns the onOptionsItemSelected() method of parent class
132 }
133
134 @SuppressLint("StatementWithEmptyBody")
135 @Override
136 public boolean onNavigationItemSelected(MenuItem item) {
137   int id = item.getItemId();
138   if (id == R.id.nav_add_student) { // if id matches with id of add student option
139     Intent intent = new Intent(getApplicationContext(), AddStudentActivity.class); // defining the next activity and go to next activity
140     startActivity(intent);

```

Line 149, Column 76      Tab Size: 4      Java

```

-/Desktop/facetedet/M.MainActivity.java - Sublime Text (UNREGISTERED)
121     @Override
122     public boolean onOptionsItemSelected(MenuItem item) {           // this functions overrides the function defined in the parent class "Menu".
123         // Handle action bar item clicks here. The action bar will
124         // automatically handle clicks on the Home/Up button, so long
125         int id = item.getItemId();                                // get the id of the selected item
126         if (id == R.id.action_settings) {                         // noinspection SimplifiableIfStatement
127             return true;
128         }
129         return super.onOptionsItemSelected(item);                // returns the onOptionsItemSelected() method of parent class
130     }
131
132     @SuppressWarnings("StatementWithEmptyBody")
133     @Override
134     public boolean onNavigationItemSelected(MenuItem item) {    // suppresses warning specially for low level api android versions
135         int id = item.getItemId();                                // this functions overrides the function defined in the parent class "Menu".
136         // this method handles navigation view item-clicks here.
137         // get the id of clicked option
138         if (id == R.id.nav_add_student) {                         // if id matches with id of add student option
139             Intent intent = new Intent(getApplicationContext(),AddStudentActivity.class);
140             startActivity(intent);                                // define the next activity and go to next activity.
141         } else if (id == R.id.nav_edit_student) {                 // if id matches with id of edit student option
142             Intent intent = new Intent(getApplicationContext(),EditStudentActivity.class);
143             startActivity(intent);                                // define the next activity and go to next activity.
144         } else if (id == R.id.nav_view_student) {                  // if id matches with id of view student option
145             Intent intent = new Intent(getApplicationContext(),ViewStudentActivity.class);
146             startActivity(intent);                                // define the next activity and go to next activity.
147         } else if (id == R.id.nav_delete_student) {               // if id matches with id of delete student option
148             Intent intent = new Intent(getApplicationContext(),DeleteStudentActivity.class);
149             startActivity(intent);                                // define the next activity and go to next activity.
150         } else if (id == R.id.nav_logout) {                        // if id matches with id of logout option
151             new AlertDialog.Builder(this)                         // launches new alert dialog box
152                 .setIcon(android.R.drawable.ic_dialog_alert)   // sets the icon for dialog box
153                 .setTitle("Log-out")                          // sets the title for the dialog box
154                 .setMessage("Are you sure you want to logout?") // sets the message to be displayed
155                 .setPositiveButton("Yes", new DialogInterface.OnClickListener() // defines tasks to be performed when user clicks yes.
156                 {
157                     @Override
158                     public void onClick(DialogInterface dialog, int which) { // specifies the task to be performed when yes is clicked
159                         Intent intent = new Intent(MainActivity.this, LoginActivity.class);
160                         startActivity(intent);                            // define the next activity and go to next activity.
161                         finish();                                     // finish this activity
162                     }
163                 })
164                 .setNegativeButton("No", null)                      // if the user selects no, then do nothing
165                 .show();
166         }
167
168         DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout); // find the drawer for controlling
169         drawer.closeDrawer(GravityCompat.START);                                // this closes the drawer
170         return true;                                                       // return true denoting operation is complete
171     }
172 }
173

```

Line 149, Column 76

Tab Size: 4 Java

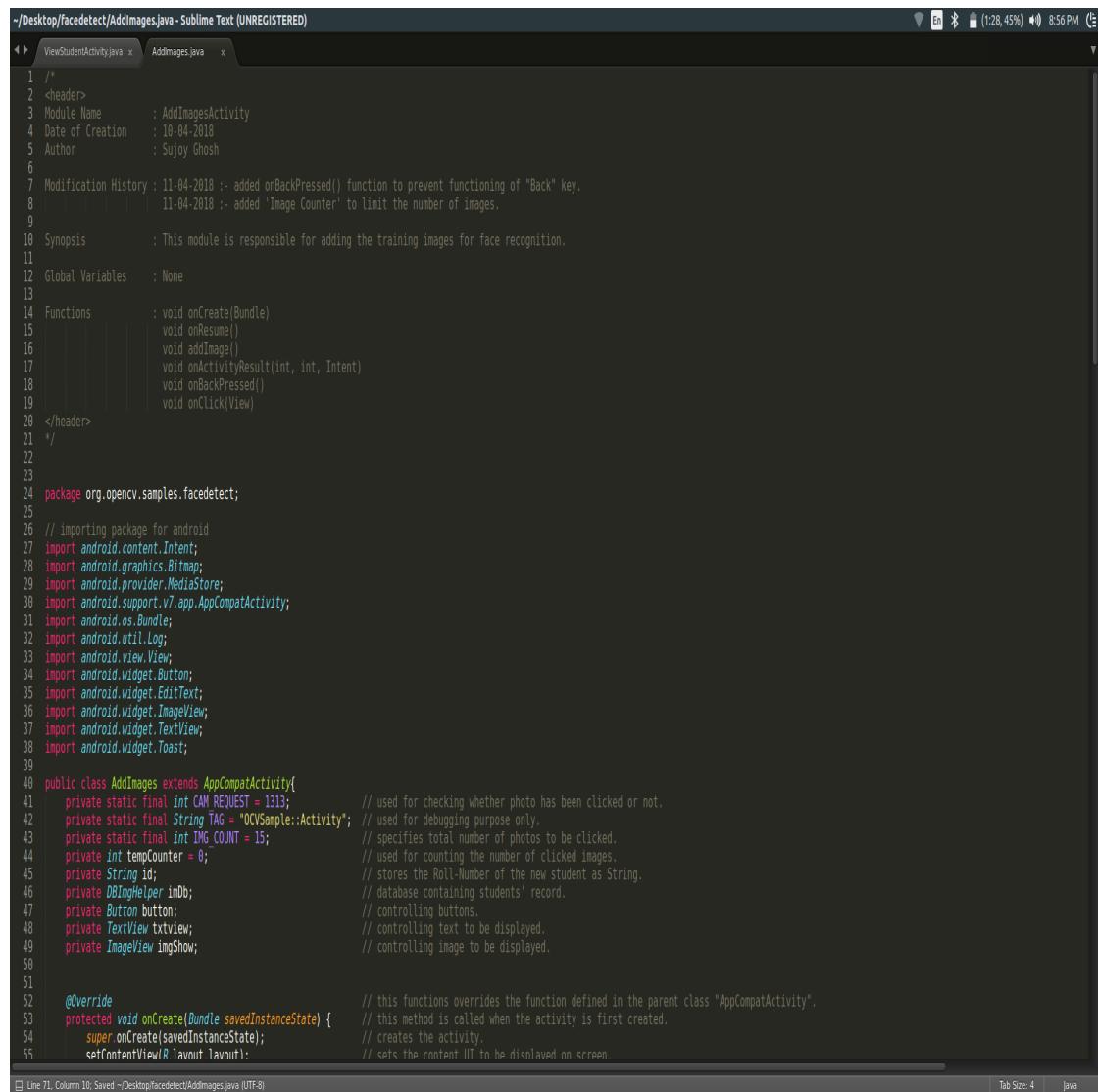
## 4 Add Student

```
-Desktop/facdetect/AddStudentActivity.java - Sublime Text (UNREGISTERED)
```

```
34
35     private DBHelper myDb;           // database containing students' records.
36     private EditText editName;       // textBox containing name of the student.
37     private EditText editSurname;    // textBox containing surname of the student.
38     private EditText editCourse;    // textBox containing course id of the student.
39     private EditText editTxtId;     // textBox containing roll number of the student.
40     private Button btnAddData;      // controlling button.
41     private String Id;             // stores the roll number of the student.
42
43     @Override
44     protected void onCreate(Bundle savedInstanceState) {           // this functions overrides the function defined in the parent class "AppCompatActivity".
45         super.onCreate(savedInstanceState);                         // this method is called when the activity is first created.
46         setContentView(R.layout.activity_add_student);           // creates the activity.
47         myDb = new DBHelper(this);                             // sets the content UI to be displayed on screen.
48         editName = (EditText)findViewById(R.id.editText_name);   // creating instance of the students' database.
49         editSurname = (EditText)findViewById(R.id.editText_surname); // creating instance of EditText object.
50         editCourse = (EditText)findViewById(R.id.editText_Course); // creating instance of EditText object.
51         editTxtId = (EditText)findViewById(R.id.editText_id);     // creating instance of EditText object.
52         btnAddData = (Button)findViewById(R.id.button_add);      // creating instance of Button object.
53         id = editTxtId.getText().toString();                     // storing the roll number of the corresponding student.
54         // Log.d("special", id);
55         AddData();                                         // class this method to add data.
56     }
57
58     public void AddData() {           // this method handles the data inserting into database.
59         btnAddData.setOnClickListener(                         // data inserting and new process starts if the button is clicked.
60             new View.OnClickListener() {                      // this creates the constructor for onClick().
61                 @Override
62                 public void onClick(View v) {                // this functions overrides the function defined in the parent class "AppCompatActivity".
63                     boolean isInteger = false;               // this function defines the tasks to be done when button is clicked.
64                     String courseID = editCourse.getText().toString(); // stores true if entered roll number is integer, otherwise false. initialising it here.
65                     String name = editName.getText().toString(); // fetch course id from text-field
66                     String surname = editSurname.getText().toString(); // fetch name from text-field
67                     String rollID = editTxtId.getText().toString(); // fetch surname from text-field
68                     int roll_number=0;                            // fetch roll id from text-field
69                     try {                                     // initialising roll number
70                         roll_number = Integer.parseInt(editTxtId.getText().toString()); // error handling when entered roll number is not integer
71                         // takes the data entered in textBox and parses it to integer, if possible.
72                         if (roll_number > 0) {                   // takes the data entered in textBox and parses it to integer, if possible.
73                             isInteger = true;                  // assign true to the variable if it is positive
74                         }
75                     } catch ( NumberFormatException e ) {        // if there is an error, with entered roll number not an integer, handle the case separately
76                         isInteger = false;                    // assign false to the variable.
77                     }
78
79                     // if any field is empty, display the same
80                     if (courseID.isEmpty() || surname.isEmpty() || name.isEmpty() || rollID.isEmpty()) {
81                         Toast.makeText(getApplicationContext(), "Field(s) is empty", Toast.LENGTH_LONG).show();
82                     } else if(isInteger) {                      // if the entered value is integer.
83                         // insert the values into database iff the roll number is not present in same course.
84                         if(myDb.ifStudentExists(roll_number, courseID) == Boolean.FALSE) {
85                             boolean isInserted = myDb.insertData(roll_number,
86                                 editName.getText().toString(),
87                                 editSurname.getText().toString(),
88                                 editCourse.getText().toString());
89                         }
90                     }
91                 }
92             }
93         }
94     }
95
96     Tab Size: 4
```

```
~/Desktop/facetedetect/AddStudentActivity.java - Sublime Text (UNREGISTERED)
AddStudentActivity.java x
16     isInteger = false;                                // assign false to the variable.
17 }
18
19     // if any field is empty, display the same
20     if (courseID.isEmpty() || surname.isEmpty() || name.isEmpty() || rollID.isEmpty()) {
21         Toast.makeText(AddStudentActivity.this, "Field(s) is empty", Toast.LENGTH_LONG).show();
22     } else if(isInteger) {                            // if the entered value is integer.
23         // insert the values into database iff the roll number is not present in same course.
24         if(myDb.ifStudentExists(roll_number, courseID) == Boolean.FALSE) {
25             boolean isInserted = myDb.insertData( roll_number,
26                     editText.getText().toString(),
27                     editSurname.getText().toString(),
28                     editCourse.getText().toString() );
29             if(isInserted == true) {          // if successfully inserted into database.
30                 // display message onto screen regarding confirmation of data entry.
31                 Toast.makeText(AddStudentActivity.this, "Data Inserted", Toast.LENGTH_LONG).show();
32                 // this defines the next activity to start if data have been added.
33                 Intent intent = new Intent(AddStudentActivity.this, AddImages.class);
34                 // stores the roll number to be used in next activity.
35                 intent.putExtra("id", editText.getText().toString());
36                 startActivity(intent);        // starts the next activity/opens the next screen view.
37                 finish();                      // finishes this activity, cleaning all data and memory.
38             } else {                        // if the roll number is already present in the database with same course id
39                 // display message onto screen informing the same.
40                 Toast.makeText(AddStudentActivity.this,"Data not Inserted",Toast.LENGTH_LONG).show();
41             }
42         } else {                          // if the entered roll number is not an integer.
43             // display message onto screen informing the same.
44             Toast.makeText(AddStudentActivity.this,"Roll Number is not integer",Toast.LENGTH_LONG).show();
45         }
46     }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
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 }
```

## 5 Add Images



The screenshot shows a Sublime Text window with two tabs: "ViewStudentActivity.java" and "AddImages.java". The "AddImages.java" tab is active, displaying Java code for an Android application. The code includes a header section with module details, imports for various Android components like Intent, Bitmap, and TextView, and a class definition for AddImages extending AppCompatActivity. The class contains fields for a database helper, button, text view, and image view, along with a constructor that calls super.onCreate(). The code is annotated with comments explaining the purpose of each variable and method.

```
1 /*
2 <header>
3 Module Name      : AddImagesActivity
4 Date of Creation : 10-04-2018
5 Author           : Sujoy Ghosh
6
7 Modification History : 11-04-2018 :: added onBackPressed() function to prevent functioning of "Back" key.
8                   : 11-04-2018 :: added 'Image Counter' to limit the number of images.
9
10 Synopsis        : This module is responsible for adding the training images for face recognition.
11
12 Global Variables : None
13
14 Functions       : void onCreate(Bundle)
15             void onResume()
16             void addImage()
17             void onActivityResult(int, int, Intent)
18             void onBackPressed()
19             void onClick(View)
20 </header>
21 */
22
23
24 package org.opencv.samples.facedetect;
25
26 // importing package for android
27 import android.content.Intent;
28 import android.graphics.Bitmap;
29 import android.provider.MediaStore;
30 import android.support.v7.app.AppCompatActivity;
31 import android.os.Bundle;
32 import android.util.Log;
33 import android.view.View;
34 import android.widget.Button;
35 import android.widget.EditText;
36 import android.widget.ImageView;
37 import android.widget.TextView;
38 import android.widget.Toast;
39
40 public class AddImages extends AppCompatActivity{
41     private static final int CAM_REQUEST = 1313;          // used for checking whether photo has been clicked or not.
42     private static final String TAG = "OCVSample::Activity"; // used for debugging purpose only.
43     private static final int IMG_COUNT = 15;              // specifies total number of photos to be clicked.
44     private int tempCounter = 0;                          // used for counting the number of clicked images.
45     private String id;                                  // stores the Roll-Number of the new student as String.
46     private DBImgHelper inDb;                           // database containing students' record.
47     private Button button;                             // controlling buttons.
48     private TextView txtview;                         // controlling text to be displayed.
49     private ImageView imgShow;                        // controlling image to be displayed.
50
51
52     @Override
53     protected void onCreate(Bundle savedInstanceState) { // this functions overrides the function defined in the parent class "AppCompatActivity".
54         super.onCreate(savedInstanceState); // this method is called when the activity is first created.
55         setContentView(R.layout.activity_main); // creates the activity.
56         // sets the content ill to be displayed on screen.
57     }
58 }
```

```

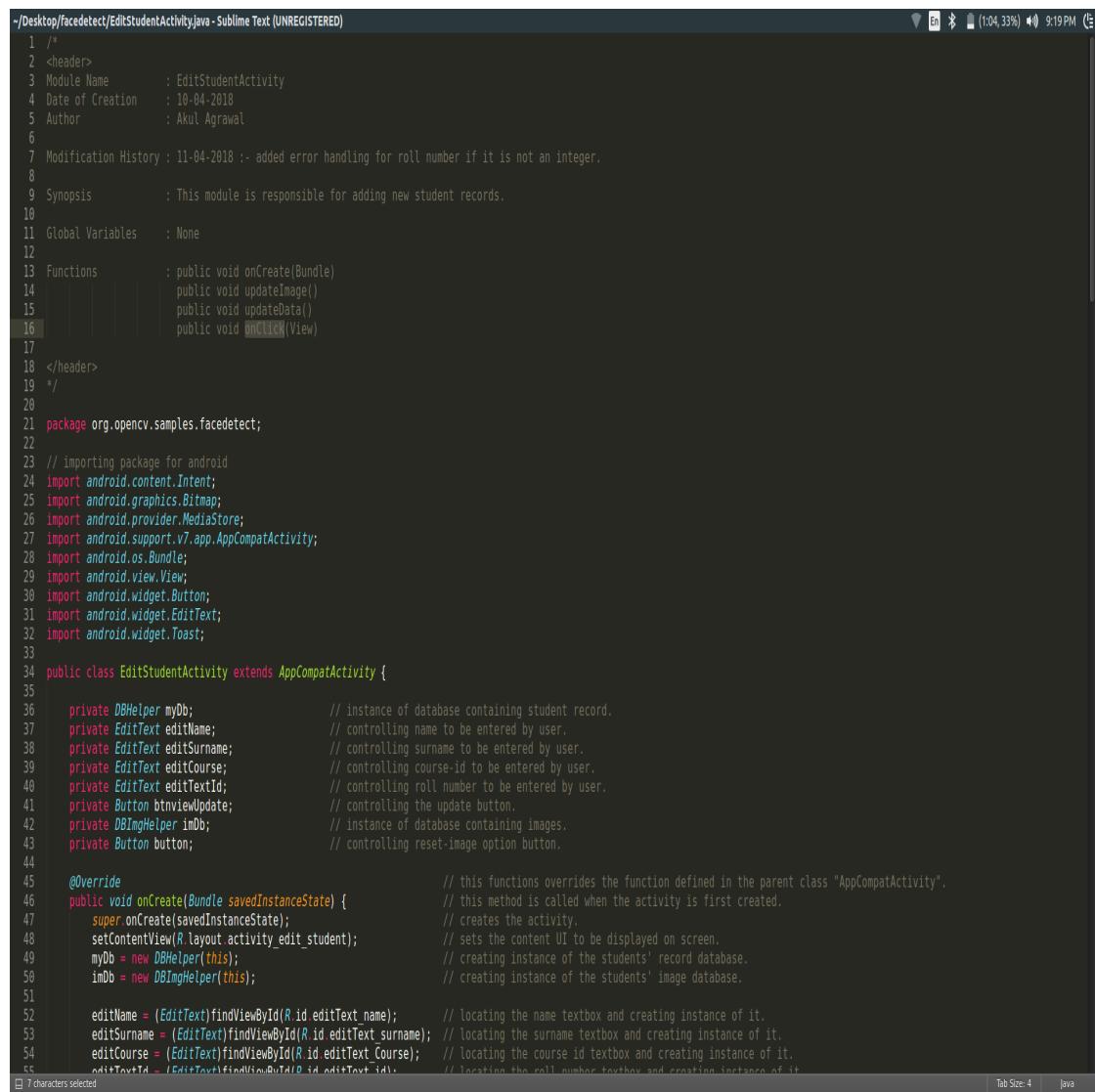
~/Desktop/facdetec/AddImages.java - Sublime Text (UNREGISTERED)
ViewStudentActivity.java AddImages.java

52     @Override
53     protected void onCreate(Bundle savedInstanceState) { // this functions overrides the function defined in the parent class "AppCompatActivity".
54         super.onCreate(savedInstanceState); // this method is called when the activity is first created.
55         setContentView(R.layout.layout); // sets the content UI to be displayed on screen.
56         id = getIntent().getStringExtra("id"); // fetches the roll number of the corresponding student for whom images are to be added.
57         tempCounter = 0; // initialising tempCounter.
58         imDb = new DBImageHelper(this); // creating instance of the students' database.
59         imgShow = (ImageView) findViewById(R.id.showImg); // creating instance of ImageView object.
60         button = (Button) findViewById(R.id.captureImg); // creating instance of Button object.
61         txtview = (TextView) findViewById(R.id.imagesLeft); // creating instance of TextView object.
62     }
63
64     @Override // this functions overrides the function defined in the parent class "AppCompatActivity".
65     public void onResume() { // this method is called when the activity is running.
66         super.onResume(); // creates the activity.
67         if (tempCounter != IMG_COUNT) { // checks whether number of photos equal to total photo to be closed.
68             addImage(); // if not, then call the addImage() to add new image
69         } else { // else, if they are equal
70             button.setText("Return to Main Menu"); // Modify the displayed text of Button.
71         }
72         if (tempCounter == IMG_COUNT) { // if number of photos equal to total photo to be closed.
73             button.setOnClickListener( // new process starts if the button is clicked.
74                 new View.OnClickListener() { // this creates the constructor for onClick().
75                     @Override // overrides the default function defined in the parent class "AppCompatActivity".
76                     public void onClick(View v) { // this function defines the tasks to be done when button is clicked.
77                         Intent nextPage = new Intent(AddImages.this, // this defines the next activity to start if images have been added.
78                                         MainActivity.class); // MainActivity class);
79                         startActivity(nextPage); // starts the next activity/opens the next screen view.
80                         finish(); // finishes this activity, cleaning all data and memory.
81                     }
82                 });
83         }
84     }
85
86     public void addImage() { // this method is responsible for taking new image.
87         button.setOnClickListener( // if Button is clicked, then it calls the camera for taking photo.
88             new AddImages.btnTakePhotoClicker()); // Camera is being called using this method.
89     }
90 }
91
92     @Override // overrides the default function defined in the parent class "AppCompatActivity".
93     protected void onActivityResult(int requestCode, int resultCode, Intent data) { // this method is responsible for accepting images and storing them in the database.
94         super.onActivityResult(requestCode, resultCode, data); // this method is called when the activity is running.
95
96         if(requestCode == CAM_REQUEST) { // if the photo has been accepted by the user.
97             Bitmap bitmap = (Bitmap) data.getExtras().get("data"); // this fetches the photo and stores into a variable.
98             boolean isInserted = imDb.insertImg(Integer.parseInt(id), bitmap); // inserts the image into database and checks whether it is properly inserted
99             if(isInserted) { // if it is properly inserted.
100                 tempCounter++; // increment tempCounter.
101                 imgShow.setImageBitmap(bitmap); // image is being displayed onto screen.
102                 int photoLeft = IMG_COUNT - tempCounter; // calculates the number of photo left to be inserted.
103                 txtview.setText(Integer.toString(photoLeft)); // displays the number of photo left to be clicked onto screen.
104                 Toast.makeText(AddImages.this, "Image Inserted".Toast.LENGTH_LONG).show(); // displays acceptance message.
105             }
106         }

```

```
/Desktop/facetedet/AddImages.java - Sublime Text (UNREGISTERED)
ViewStudentActivity.java x AddImages.java x
14 // overrides the default function defined in the parent class "AppCompatActivity".
15 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
16     super.onActivityResult(requestCode, resultCode, data);
17
18     if(requestCode == CAM_REQUEST) {
19         Bitmap bitmap = (Bitmap) data.getExtras().get("data");
20         boolean isInserted = imDb.insertImg(Integer.parseInt(id), bitmap);
21         if(isInserted) {
22             tempCounter++;
23             imgShow.setImageBitmap(bitmap);
24             int photoLeft = IMG_COUNT - tempCounter;
25             txtview.setText(Integer.toString(photoLeft));
26             Toast.makeText(AddImages.this, "Image Inserted", Toast.LENGTH_LONG).show();
27         } else {
28             Toast.makeText(AddImages.this, "Image not Inserted", Toast.LENGTH_LONG).show();
29         }
30     }
31
32     class btnTakePhotoClicker implements Button.OnClickListener {
33         @Override
34         public void onClick(View view) {
35             Intent isPhotoAcceptable = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
36             startActivityForResult(isPhotoAcceptable, CAM_REQUEST);
37         }
38     }
39
40     @Override
41     public void onBackPressed() {
42         // overrides the default function defined in the parent class "AppCompatActivity".
43         // this method is responsible for not letting the user going back without completing the required number of images
44         // this shows the text if someone presses the back button.
45         Toast.makeText(AddImages.this, "You cannot go back until the process completes", Toast.LENGTH_LONG).show();
46     }
47 }
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
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
```

## 6 Edit Student



The screenshot shows a Sublime Text editor window with the following details:

- File Path: ~/Desktop/facedetect/EditStudentActivity.java - Sublime Text (UNREGISTERED)
- File Type: Java
- Code Content:

```
1  /*
2  <header>
3  Module Name      : EditStudentActivity
4  Date of Creation : 10-04-2018
5  Author           : Akul Agrawal
6
7  Modification History : 11-04-2018 :- added error handling for roll number if it is not an integer.
8
9  Synopsis         : This module is responsible for adding new student records.
10
11 Global Variables : None
12
13 Functions        : public void onCreate(Bundle)
14                  |     public void updateImage()
15                  |     public void updateData()
16                  |     public void onClick(View)
17
18 </header>
19 */
20
21 package org.opencv.samples.facedetect;
22
23 // importing package for android
24 import android.content.Intent;
25 import android.graphics.Bitmap;
26 import android.provider.MediaStore;
27 import android.support.v7.app.AppCompatActivity;
28 import android.os.Bundle;
29 import android.view.View;
30 import android.widget.Button;
31 import android.widget.EditText;
32 import android.widget.Toast;
33
34 public class EditStudentActivity extends AppCompatActivity {
35
36     private DBHelper myDb;           // instance of database containing student record.
37     private EditText editTextName;   // controlling name to be entered by user.
38     private EditText editTextSurname; // controlling surname to be entered by user.
39     private EditText editTextCourse; // controlling course-id to be entered by user.
40     private EditText editTextId;    // controlling roll number to be entered by user.
41     private Button btnviewUpdate;   // controlling the update button.
42     private DBImgHelper imDb;       // instance of database containing images.
43     private Button button;          // controlling reset-image option button.
44
45     @Override
46     public void onCreate(Bundle savedInstanceState) {           // this functions overrides the function defined in the parent class "AppCompatActivity".
47         super.onCreate(savedInstanceState);                      // this method is called when the activity is first created.
48         setContentView(R.layout.activity_edit_student);        // creates the activity.
49         myDb = new DBHelper(this);                           // sets the content UI to be displayed on screen.
50         imDb = new DBImgHelper(this);                        // creating instance of the students' record database.
51
52         editTextName = (EditText) findViewById(R.id.editText_name); // creating instance of the students' image database.
53         editTextSurname = (EditText) findViewById(R.id.editText_surname); // locating the name textbox and creating instance of it.
54         editTextCourse = (EditText) findViewById(R.id.editText_Course); // locating the surname textbox and creating instance of it.
55         editTextId = (EditText) findViewById(R.id.editText_id);        // locating the course id textbox and creating instance of it.
56     }
57 }
```
- Status Bar: Shows battery level (104, 33%), signal strength, and time (9:19 PM).
- Bottom Status: Tab Size: 4, Java.

```

-/Desktop/facetedetct/EditStudentActivity.java - Sublime Text (UNREGISTERED)
52     editName = (EditText)findViewById(R.id.editText_name);           // locating the name textbox and creating instance of it.
53     editSurname = (EditText)findViewById(R.id.editText_surname);      // locating the surname textbox and creating instance of it.
54     editCourse = (EditText)findViewById(R.id.editText_Course);        // locating the course id textbox and creating instance of it.
55     editTextId = (EditText)findViewById(R.id.editText_id);           // locating the roll number textbox and creating instance of it.
56     btviewUpdate= (Button)findViewById(R.id.button_update);          // locating the update button and creating instance of it.
57
58     button = (Button) findViewById(R.id.ImgButton);                   // locating the update image button and creating instance of it.
59
60     updateImage();                                                 // method for updating image is called when the corresponding button is clicked.
61     updateData();                                                 // method for updating record is called when the corresponding button is clicked.
62 }
63
64 public void updateImage() {
65     button.setOnClickListener(                                     // this method is responsible for updating images of the corresponding roll number.
66         new View.OnClickListener() {                           // new process starts if the button is clicked.
67             @Override                                         // this creates the constructor for onClick().
68             public void onClick(View v) {                     // this functions overrides the function defined in the parent class "AppCompatActivity".
69                 boolean isInteger = false;                  // declaring boolean variable for checking if entered roll number is integer and initialising it.
70                 String courseId = editCourse.getText().toString();
71                 String name = editName.getText().toString();
72                 String surname = editSurname.getText().toString();
73                 String rollID = editTextId.getText().toString();
74                 int roll_number=0;                          // declaring integer for storing the value of entered roll number.
75                 try {                                 // error handling for entered roll number.
76                     roll_number = Integer.parseInt(editTextId.getText().toString()); // try to parse/convert the entered roll number to integer if possible.
77                     isInteger = true;                      // if parsing is possible, then assign true to this variable.
78                 } catch ( NumberFormatException e ) {       // if the entered string cannot be converted to integer, then handle this case separately.
79                     isInteger = false;                     // if parsing is not possible, then assign false to this variable.
80                 }
81
82                 if (courseID.isEmpty() || surname.isEmpty() || name.isEmpty() || rollID.isEmpty()) {
83                     Toast.makeText(EditStudentActivity.this, "Field(s) is empty", Toast.LENGTH_LONG).show();
84                 } else if(isInteger) {
85                     if ( myDb.ifStudentExists(roll_number, courseId) ) { // if the entered value is integer.
86                         Integer deletedImg = imDb.deleteImg(roll_number); // try to delete the images of corresponding roll number.
87                         if (deletedImg > 0) {                            // if the roll number exists.
88                             Intent intent = new Intent(EditStudentActivity.this, AddImages.class); // display the message that images have been deleted.
89                             intent.putExtra("id", editTextId.getText().toString()); // this defines the next activity(add images) to start if images have been deleted.
90                             startActivity(intent);                                // stores the roll number to be used in next activity.
91                             finish();                                         // starts the next activity/opens the next screen view.
92                         } else{                                         // if nothing is deleted, display roll number is not present.
93                             Toast.makeText(EditStudentActivity.this, "Roll ID not present in database", Toast.LENGTH_LONG).show();
94                         }
95                     } else {                                         // display roll number is not present.
96                         Toast.makeText(EditStudentActivity.this, "Roll ID with given course not present", Toast.LENGTH_LONG).show();
97                     }
98                 } else{                                         // if the entered value is not integer, display the message showing the same.
99                     Toast.makeText(EditStudentActivity.this,"Entered Roll ID is not integer",Toast.LENGTH_LONG).show();
100                }
101            }
102        }
103    } else{                                         // if the entered value is not integer, display the message showing the same.
104        Toast.makeText(EditStudentActivity.this,"Entered Roll ID is not integer",Toast.LENGTH_LONG).show();
105    }
106}

```

```

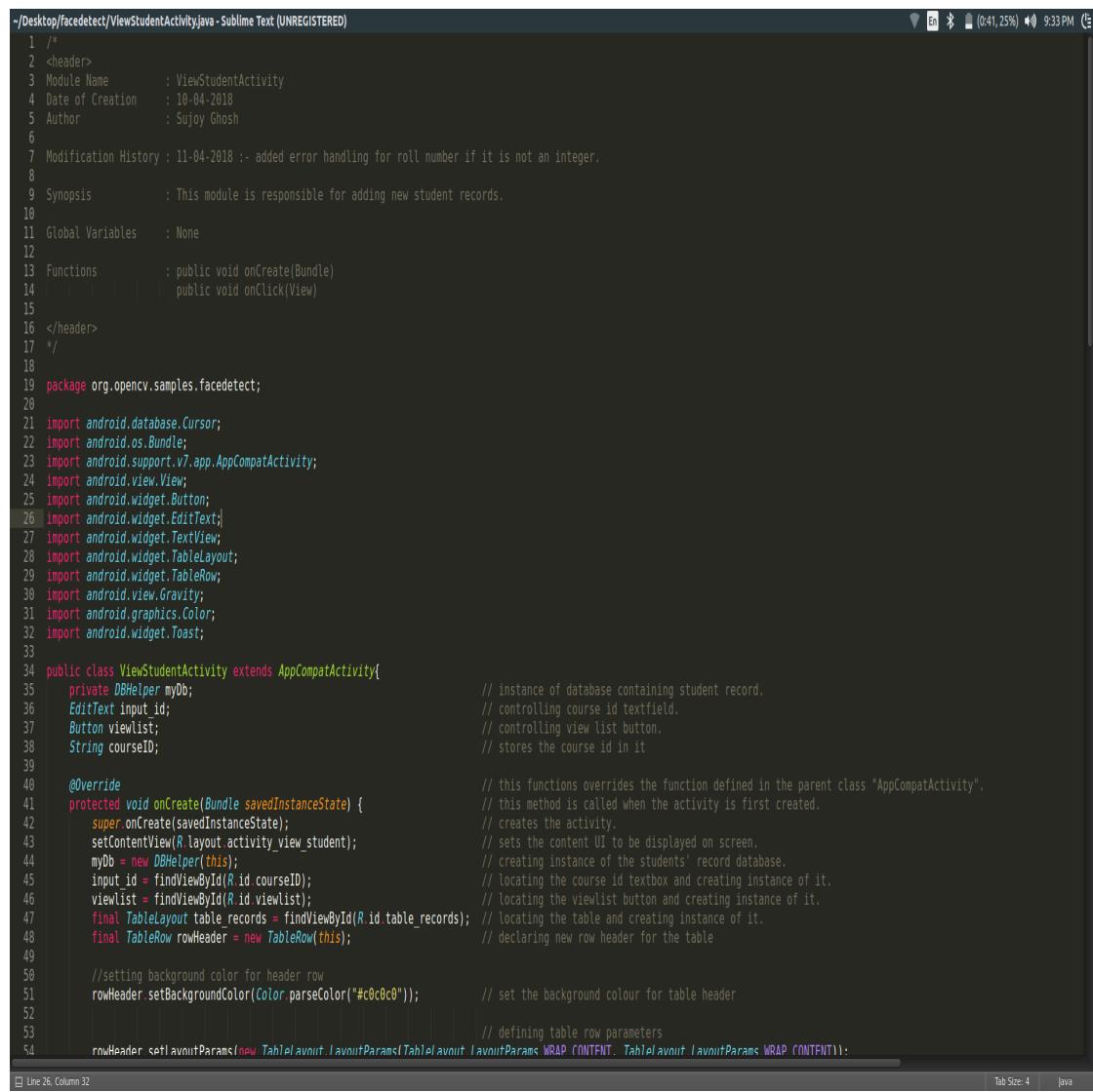
-/Desktop/facetedet/EditStudentActivity.java - Sublime Text (UNREGISTERED)
100         } else { // display roll number is not present.
101             Toast.makeText(EditStudentActivity.this, "Roll ID with given course not present", Toast.LENGTH_LONG).show();
102         }
103     } else{ // if the entered value is not integer, display the message showing the same.
104         Toast.makeText(EditStudentActivity.this,"Entered Roll ID is not integer",Toast.LENGTH_LONG).show();
105     }
106 }
107 );
108 }
109 }
110 }

111 public void updateData() {
112     btnviewUpdate.setOnClickListener(
113         new View.OnClickListener() {
114             @Override
115             public void onClick(View v) {
116                 boolean isInteger = false;
117                 String courseID = editCourse.getText().toString();
118                 String name = editName.getText().toString();
119                 String surname = editSurname.getText().toString();
120                 String rollID = editTxtId.getText().toString();
121                 int roll_number=0;
122                 try {
123                     roll_number = Integer.parseInt(editTxtId.getText().toString());
124                     isInteger = true;
125                 } catch ( NumberFormatException e ) {
126                     isInteger = false;
127                 }
128             }
129
130             if (courseID.isEmpty() || surname.isEmpty() || name.isEmpty() || rollID.isEmpty()) {
131                 Toast.makeText(EditStudentActivity.this, "Field(s) is empty", Toast.LENGTH_LONG).show();
132             } else if(isInteger) {
133                 // if the entered value is integer.
134                 boolean isUpdate = myDb.updateData(roll_number,
135                     editName.getText().toString(),
136                     editSurname.getText().toString(), editCourse.getText().toString());
137                 if (isUpdate == true) {
138                     // if update is successful
139                     Toast.makeText(EditStudentActivity.this, "Data Updated", Toast.LENGTH_LONG).show();
140                 } else {
141                     // display the message that roll number is not present.
142                     Toast.makeText(EditStudentActivity.this, "Data not Updated", Toast.LENGTH_LONG).show();
143                 }
144             } else {
145                 // if the entered value is not integer, display the message showing the same.
146                 Toast.makeText(EditStudentActivity.this,"Roll Number is not integer",Toast.LENGTH_LONG).show();
147             }
148         }
149     );
150 }
151 }
152 }
153 }
154 }

Line 135: Column 65
Tab Size: 4 Java

```

## 7 View Student List



The screenshot shows a Sublime Text editor window with the following details:

- File Path: ~/Desktop/facedetect/ViewStudentActivity.java - Sublime Text (UNREGISTERED)
- Status Bar: En (0:41, 25%) 9:33 PM (C)
- Code Content:

```
1  /*
2   <header>
3   Module Name      : ViewStudentActivity
4   Date of Creation : 10-04-2018
5   Author          : Sujoy Ghosh
6
7   Modification History : 11-04-2018 :- added error handling for roll number if it is not an integer.
8
9   Synopsis        : This module is responsible for adding new student records.
10
11  Global Variables : None
12
13  Functions       : public void onCreate(Bundle)
14  |                  | public void onClick(View)
15
16 </header>
17 */
18
19 package org.opencv.samples.facedetect;
20
21 import android.database.Cursor;
22 import android.os.Bundle;
23 import android.support.v7.app.AppCompatActivity;
24 import android.view.View;
25 import android.widget.Button;
26 import android.widget.EditText;
27 import android.widget.TextView;
28 import android.widget.TableLayout;
29 import android.widget.TableRow;
30 import android.view.Gravity;
31 import android.graphics.Color;
32 import android.widget.Toast;
33
34 public class ViewStudentActivity extends AppCompatActivity{
35     private DBHelper myDb;           // instance of database containing student record.
36     EditText input_id;              // controlling course id textfield.
37     Button viewlist;                // controlling view list button.
38     String courseId;                // stores the course id in it
39
40     @Override
41     protected void onCreate(Bundle savedInstanceState) {           // this functions overrides the function defined in the parent class "AppCompatActivity".
42         super.onCreate(savedInstanceState);                         // this method is called when the activity is first created.
43         setContentView(R.layout.activity_view_student);           // creates the activity.
44         myDb = new DBHelper(this);                             // sets the content UI to be displayed on screen.
45         input_id = findViewById(R.id.courseID);                 // creating instance of the students' record database.
46         viewlist = findViewById(R.id.viewlist);                 // locating the course id textbox and creating instance of it.
47         final TableLayout table_records = findViewById(R.id.table_records); // locating the viewlist button and creating instance of it.
48         final TableRow rowHeader = new TableRow(this);          // locating the table and creating instance of it.
49
50         //declaring new row header for the table
51         rowHeader.setBackgroundColor(Color.parseColor("#c0c0c0")); // declaring new row header for the table
52
53         //setting background color for header row
54         rowHeader.setLayoutParams(new TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT, TableLayout.LayoutParams.WRAP_CONTENT)); // set the background colour for table header
55
56         // defining table row parameters
57         rowHeader.setLayoutParams(new TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT, TableLayout.LayoutParams.WRAP_CONTENT));
58     }
59 }
```
- Bottom Status Bar: Line 26, Column 32 | Tab Size: 4 | Java

```

-/Desktop/facetedetc/ViewStudentActivity.java • Sublime Text (UNREGISTERED)
31 import android.graphics.Color;
32 import android.widget.Toast;
33
34 public class ViewStudentActivity extends AppCompatActivity{
35     private DBHelper myDb;           // instance of database containing student record.
36     EditText input_id;              // controlling course id textfield.
37     Button viewlist;                // controlling view list button.
38     String courseId;                // stores the course id in it
39
40     @Override                      // this functions overrides the function defined in the parent class "AppCompatActivity".
41     protected void onCreate(Bundle savedInstanceState) { // this method is called when the activity is first created.
42         super.onCreate(savedInstanceState); // creates the activity.
43         setContentView(R.layout.activity_view_student); // sets the content UI to be displayed on screen.
44         myDb = new DBHelper(this); // creating instance of the students' record database.
45         input_id = findViewById(R.id.courseID); // locating the course id textbox and creating instance of it.
46         viewlist = findViewById(R.id.viewlist); // locating the viewlist button and creating instance of it.
47         final TableLayout table_records = findViewById(R.id.table_records); // locating the table and creating instance of it.
48         final TableRow rowHeader = new TableRow(this); // declaring new row header for the table
49
50         //setting background color for header row
51         rowHeader.setBackgroundColor(Color.parseColor("#c0c0c0")); // set the background colour for table header
52
53         // defining table row parameters
54         rowHeader.setLayoutParams(new TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT, TableLayout.LayoutParams.WRAP_CONTENT));
55
56         // Header row text
57         String[] headerText={"Roll Number","Name","Course Code","Last Appear","Box Colour"};
58
59         for(String c:headerText){ //This sets parameters and text for five columns of table for header row
60             TextView tv = new TextView(this); // declare variable to store the column values
61             // setting height an length parameters using wrap_content params
62             tv.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.WRAP_CONTENT));
63             tv.setGravity(Gravity.CENTER); // alignment of text wrt cell
64             tv.setTextSize(18); // setting text size for header row
65             tv.setPadding(5, 5, 5, 5); // padding of cell
66             tv.setText(c); // insert the text to be displayed in column
67             rowHeader.addView(tv); // add the column into row
68         }
69         viewlist.setOnClickListener( // new process starts if the button is clicked.
70             new View.OnClickListener() { // this creates the constructor for onClick().
71                 @Override // this functions overrides the function defined in the parent class "AppCompatActivity".
72                 public void onClick(View v) { // this function defines the tasks to be done when button is clicked.
73                     if(viewlist.getText().toString().equalsIgnoreCase("View Students List")) { // if the button lable is view student, then function as table viewer
74                         courseId = input_id.getText().toString(); // fetch the course id from textfield
75                         Cursor res = myDb.getCourseData(courseId); // fetch the list of all students with given course id from the database
76                         if (courseId.isEmpty()) { // if course id is empty
77                             Toast.makeText(ViewStudentActivity.this, "Field is empty", Toast.LENGTH_LONG).show();
78                         } else if (res.getCount() == 0) { // if number of student corresponding to course id is 0, then display suitable message
79                             Toast.makeText(ViewStudentActivity.this, "No student record from selected course", Toast.LENGTH_LONG).show();
80                         } else { // add a row into table
81                             table_records.addView(rowHeader);
82                         }
83                     }
84                 }
85             }
86         );
87     }

```

Line 68, Column 10

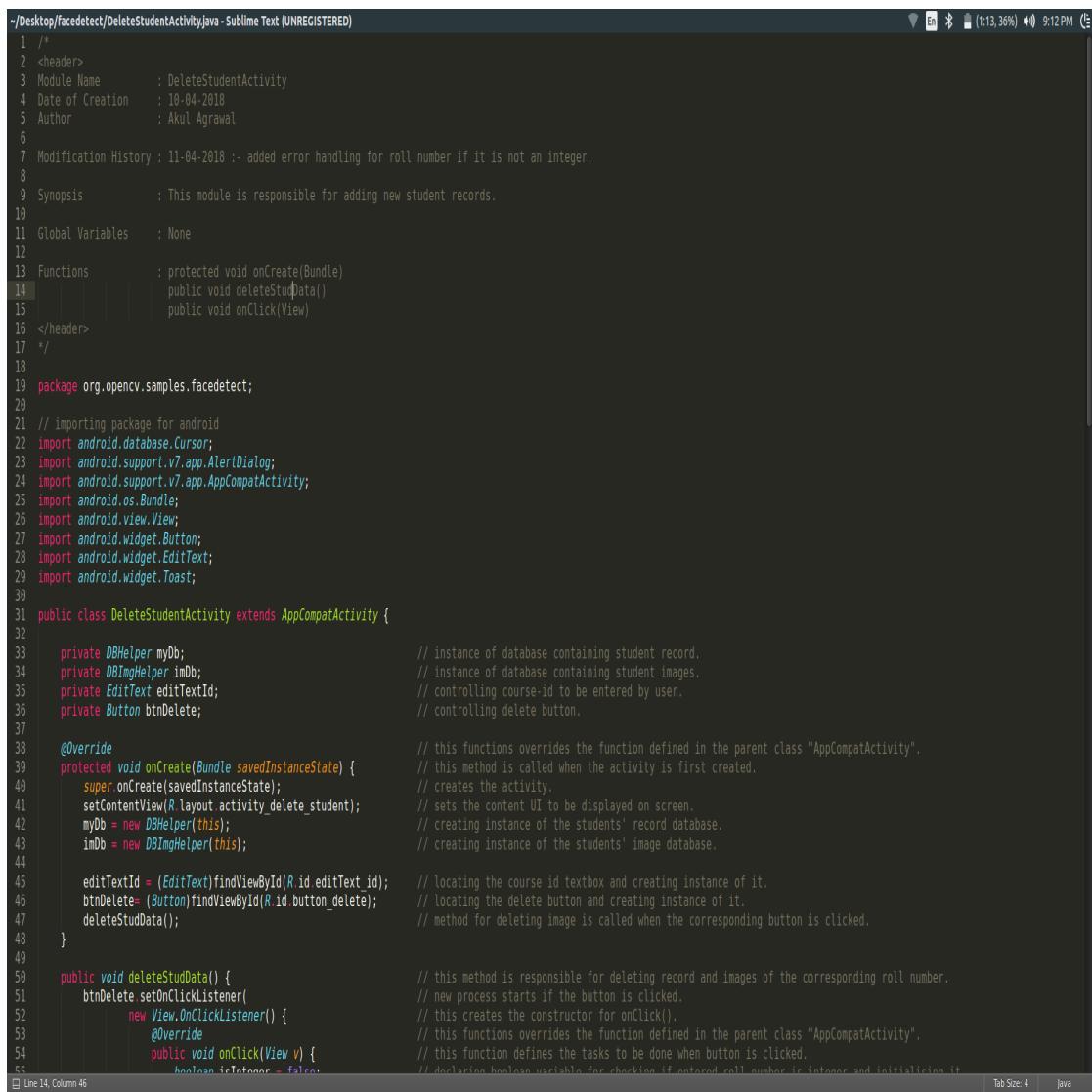
Tab Size: 4 Java

```

-/Desktop/facetedetct/ViewStudentActivity.java • Sublime Text (UNREGISTERED)
67     rowHeader.addView(tv);                                // add the column into row
68 }
69     viewlist.setOnClickListener(                         // new process starts if the button is clicked.
70         new View.OnClickListener() {                   // this creates the constructor for onClick().
71             @Override
72             public void onClick(View v) {               // this function overrides the function defined in the parent class "AppCompatActivity".
73                 if(viewlist.getText().toString().equalsIgnoreCase("View Students List")) {
74                     courseID = input_id.getText().toString(); // fetch the course id from textfield
75                     Cursor res = mydb.getCourseData(courseID); // fetch the list of all students with given course id from the database
76                     if(courseID.isEmpty()) {
77                         Toast.makeText(ViewStudentActivity.this, "Field is empty", Toast.LENGTH_LONG).show();
78                     } else if (res.getCount() == 0) {           // if number of student corresponding to course id is 0, then display suitable message
79                         Toast.makeText(ViewStudentActivity.this, "No student record from selected course", Toast.LENGTH_LONG).show();
80                     } else {                                // show message
81                         String[] colText = {res.getString(0), res.getString(1), res.getString(3), res.getString(4), res.getString(5)};
82                         TableRow row = new TableRow(ViewStudentActivity.this);
83                         row.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.WRAP_CONTENT));
84                         String[] colText = {res.getString(0), res.getString(1), res.getString(3), res.getString(4), res.getString(5)};
85                         for (String text : colText) {           //This sets parameters and text for the columns of table for header row
86                             TextView tv = new TextView(ViewStudentActivity.this);          // declare variable to store the column values
87                             tv.setText(text);                                         // this stores the column value
88                             tv.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.WRAP_CONTENT)); //setting height an length
89                             tv.setGravity(Gravity.CENTER);                           // alignment of text wrt cell
90                             tv.setTextSize(16);                                    // setting text size for header row
91                             tv.setPadding(5, 5, 5);                                // padding of cell
92                             tv.setText(text);                                     // insert the text to be displayed in column
93                             row.addView(tv);                                    // add the column into row
94                         }
95                         table_records.addView(row);                          // add the row into table
96                         viewlist.setText("Clear");                         // make button label as clear
97                     }
98                 } else {                                // remove all table rows
99                     table_records.removeAllViews();                // change the text of button to "view student list" to enable view other courses
100                     viewlist.setText("View Students List");          // clear the textfield
101                     input_id.setText("");
102                 }
103             }
104         }
105     );
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 );
116 }
117 }
118 }
119 }

```

## 8 Delete Student



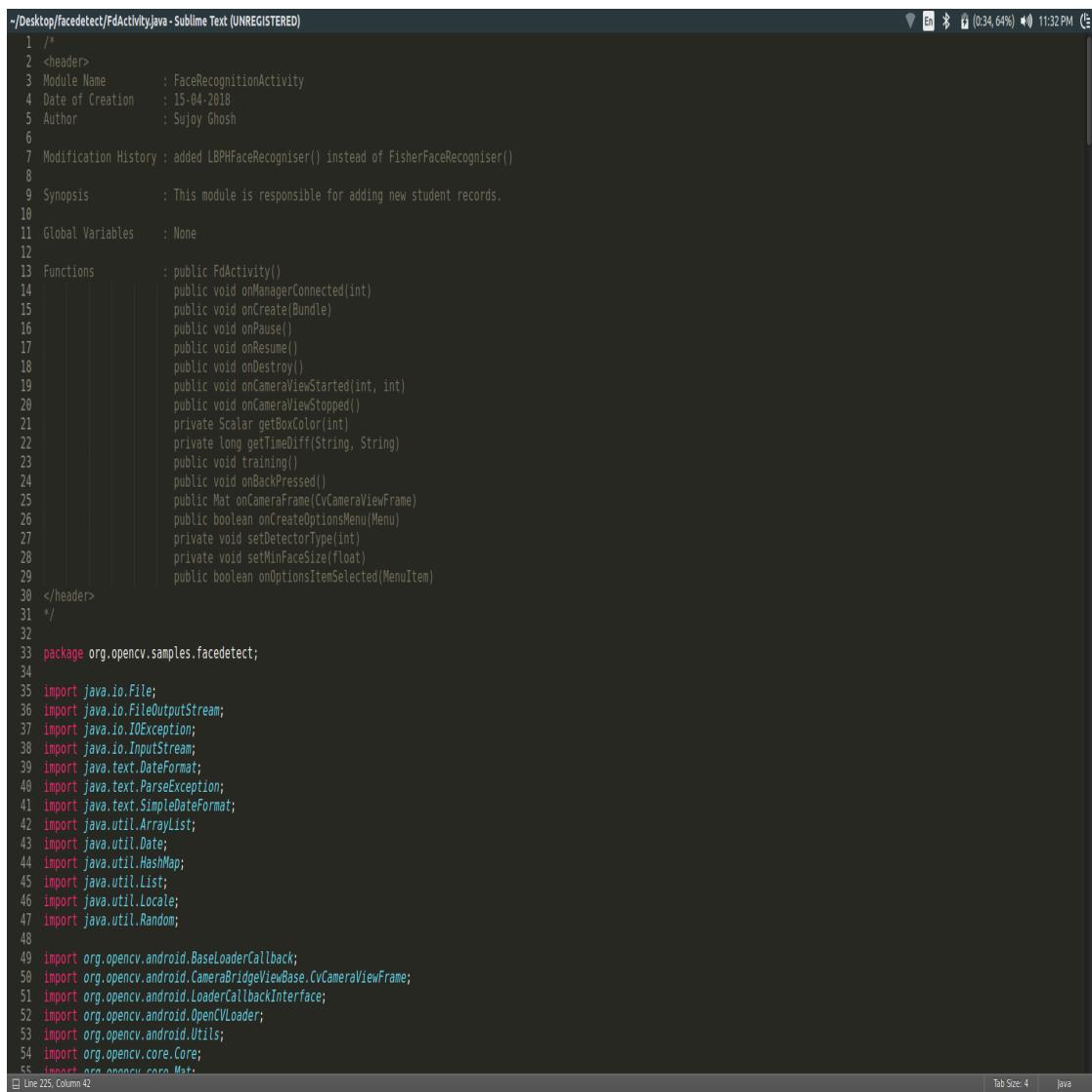
The screenshot shows a Sublime Text editor window with the following details:

- File Path: ~/Desktop/facedetect/DeleteStudentActivity.java - Sublime Text (UNREGISTERED)
- File Type: Java
- Line Number: Line 14, Column 46
- Text Size: Tab Size: 4
- Language: Java

```
1 /*
2 <header>
3 Module Name      : DeleteStudentActivity
4 Date of Creation : 10-04-2018
5 Author          : Akul Agrawal
6
7 Modification History : 11-04-2018 :- added error handling for roll number if it is not an integer.
8
9 Synopsis        : This module is responsible for adding new student records.
10
11 Global Variables : None
12
13 Functions       : protected void onCreate(Bundle)
14 |                 public void deleteStudData()
15 |                 public void onClick(View)
16 </header>
17 */
18
19 package org.opencv.samples.facedetect;
20
21 // importing package for android
22 import android.database.Cursor;
23 import android.support.v7.app.AlertDialog;
24 import android.support.v7.app.AppCompatActivity;
25 import android.os.Bundle;
26 import android.view.View;
27 import android.widget.Button;
28 import android.widget.EditText;
29 import android.widget.Toast;
30
31 public class DeleteStudentActivity extends AppCompatActivity {
32
33     private DBHelper myDb;           // instance of database containing student record.
34     private DBImgHelper imDb;        // instance of database containing student images.
35     private EditText editTxtId;      // controlling course-id to be entered by user.
36     private Button btnDelete;        // controlling delete button.
37
38     @Override
39     protected void onCreate(Bundle savedInstanceState) {
40         super.onCreate(savedInstanceState);
41         setContentView(R.layout.activity_delete_student);
42         myDb = new DBHelper(this);
43         imDb = new DBImgHelper(this);
44
45         editTxtId = (EditText)findViewById(R.id.editText_id);
46         btnDelete= (Button)findViewById(R.id.button_delete);
47         deleteStudData();
48     }
49
50     public void deleteStudData() {
51         btnDelete.setOnClickListener(
52             new View.OnClickListener() {
53                 @Override
54                 public void onClick(View v) {
55                     boolean isInteger = false;
```

```
/Desktop/facdetec/DeleteStudentActivity.java - Sublime Text (UNREGISTERED)
46     btnDelete= (Button)findViewById(R.id.button_delete);           // locating the delete button and creating instance of it.
47     deleteStudData();                                              // method for deleting image is called when the corresponding button is clicked.
48 }
49
50 public void deleteStudData() {                                         // this method is responsible for deleting record and images of the corresponding roll number.
51     btnDelete.setOnClickListener(                                     // new process starts if the button is clicked.
52         new View.OnClickListener() {                                // this creates the constructor for onClick().
53             @Override                                            // this functions overrides the function defined in the parent class "AppCompatActivity".
54             public void onClick(View v) {                         // this function defines the tasks to be done when button is clicked.
55                 boolean isInteger = false;                      // declaring boolean variable for checking if entered roll number is integer and initialising it.
56                 String rollID = editTextId.getText().toString(); // fetch roll id from text-field
57                 int roll_number=0;                             // declaring integer for storing the value of entered roll number.
58                 try {                                       // error handling for entered roll number.
59                     roll_number = Integer.parseInt(editTextId.getText().toString()); // try to parse/convert the entered roll number to integer if possible.
60                     if (roll_number > 0) {                      // assign true to the variable if it is positive
61                         isInteger = true;
62                     }
63                 } catch ( NumberFormatException e ) {        // if the entered string cannot be converted to integer, then handle this case separately.
64                     isInteger = false;                         // if parsing is not possible, then assign false to this variable.
65                 }
66             }
67             // if any field is empty, display the same
68             if (rollID.isEmpty()) {
69                 Toast.makeText(DeleteStudentActivity.this, "Field(s) is empty", Toast.LENGTH_LONG).show();
70             } else if(isInteger) {                           // if the entered value is integer.
71                 Integer deletedRows = myDb.deleteData(roll_number); // try to delete the record of corresponding roll number.
72                 Integer deletedImg = jDb.deleteImg(roll_number); // try to delete the images of corresponding roll number.
73                 if (deletedRows > 0 && deletedImg > 0){          // if the roll number exists.
74                     // display the message that images have been deleted.
75                     Toast.makeText(DeleteStudentActivity.this, "Data Deleted", Toast.LENGTH_LONG).show();
76                 } else{                                         // if not successful.
77                     // display the message that roll number is not present.
78                     Toast.makeText(DeleteStudentActivity.this, "Roll ID not present in database", Toast.LENGTH_LONG).show();
79                 }
80             } else{                                         // if the entered value is not integer, display the message showing the same.
81                 Toast.makeText(DeleteStudentActivity.this,"Entered Roll ID is not integer",Toast.LENGTH_LONG).show();
82             }
83         }
84     );
85 }
86 }
87
88 }
89
```

# 9 Camera Session



The screenshot shows a Sublime Text editor window with the following details:

- File Path: ~/Desktop/facedetect/FdActivity.java
- Text: UNREGISTERED
- Status Bar: En ⚡ (0:34, 64%) 11:32 PM
- Code Content:

```
1  /*
2  <header>
3  Module Name      : FaceRecognitionActivity
4  Date of Creation : 15-04-2018
5  Author           : Sujoy Ghosh
6
7  Modification History : added LBPHFaceRecogniser() instead of FisherFaceRecogniser()
8
9  Synopsis         : This module is responsible for adding new student records.
10
11 Global Variables : None
12
13 Functions        : public FdActivity()
14                 public void onManagerConnected(int)
15                 public void onCreate(Bundle)
16                 public void onPause()
17                 public void onResume()
18                 public void onDestroy()
19                 public void onCameraViewStarted(int, int)
20                 public void onCameraViewStopped()
21                 private Scalar getBoxColor(int)
22                 private long getTimeDiff(String, String)
23                 public void training()
24                 public void onBackPressed()
25                 public Mat onCameraFrame(CvCameraViewFrame)
26                 public boolean onCreateOptionsMenu(Menu)
27                 private void setDetectorType(int)
28                 private void setMinFaceSize(float)
29                 public boolean onOptionsItemSelected(MenuItem)
30
31 </header>
32 */
33
34 package org.opencv.samples.facedetect;
35
36 import java.io.File;
37 import java.io.FileOutputStream;
38 import java.io.IOException;
39 import java.io.InputStream;
40 import java.text.ParseException;
41 import java.text.SimpleDateFormat;
42 import java.util.ArrayList;
43 import java.util.Date;
44 import java.util.HashMap;
45 import java.util.List;
46 import java.util.Locale;
47 import java.util.Random;
48
49 import org.opencv.android.BaseLoaderCallback;
50 import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
51 import org.opencv.android.LoaderCallbackInterface;
52 import org.opencv.android.OpenCVLoader;
53 import org.opencv.android.Utils;
54 import org.opencv.core.Core;
55 import org.opencv.core.Mat;
```
- Bottom Status: Line 225, Column 42 | Tab Size: 4 | Java

```
-/Desktop/facedetect/FdActivity.java - Sublime Text (UNREGISTERED)
55 import org.opencv.core.Mat;
56 import org.opencv.core.MatOfInt;
57 import org.opencv.core.MatOfRect;
58 import org.opencv.core.Point;
59 import org.opencv.core.Rect;
60 import org.opencv.core.Scalar;
61 import org.opencv.core.Size;
62 import org.opencv.android.CameraBridgeViewBase;
63 import org.opencv.android.CameraBridgeViewBase.CvCameraViewListener2;
64 import org.opencv.face.FaceRecognizer;
65 import org.opencv.objdetect.CascadeClassifier;
66 import org.opencv.imgproc.Imgproc;
67 import org.opencv.face.LBPHFaceRecognizer;
68 import android.app.Activity;
69 import android.content.Context;
70 import android.database.Cursor;
71 import android.graphics.Bitmap;
72 import android.os.Bundle;
73 import android.util.Log;
74 import android.view.Menu;
75 import android.view.MenuItem;
76 import android.view.WindowManager;
77 import android.widget.Toast;
78
79 public class FdActivity extends Activity implements CvCameraViewListener2 {
80
81     private static final String TAG = "OCVSample::Activity"; // for debugging purposes
82     public static final int JAVA_DETECTOR = 0; // for specifying it is java detector
83     public static final int NATIVE_DETECTOR = 1; // for specifying it is native/c++ based detector
84
85     private MenuItem mItemFace50; // specifies menuitem 50%
86     private MenuItem mItemFace40; // specifies menuitem 40%
87     private MenuItem mItemFace30; // specifies menuitem 30%
88     private MenuItem mItemFace20; // specifies menuitem 20%
89     private MenuItem mItemType; // specifies detector type
90     private Mat mRgba; // stores rgba form of camera view
91     private Mat mGray; // stores gray form of camera view
92     private File mCascadeFile; // used for loading cascade file
93     private CascadeClassifier mJavaDetector; // specifies java detector
94     private DetectionBasedTracker mNativeDetector; // specifies c++ detector
95
96     private int mDetectorType = JAVA_DETECTOR; // initialise detector type
97     private String[] mDetectorName; // stores possible detectors
98
99     private float mRelativeFaceSize = 0.2f; // initialise relative face size
100    private int mAbsoluteFaceSize = 0; // specifies absolute face size
101
102    private CameraBridgeViewBase mOpenCvCameraView; // for connecting opencv with camera
103    String id; // store the roll id
104    ArrayList<Mat> sourceImages; // stores the images
105    List<Integer> personIndexList; // stores indices of person
106    List<Integer> namesIntList; // stores roll number of person
107    HashMap<Integer, Integer> boundaryBoxColor; // stores box color of person
108    HashMap<Integer, String> personName; // stores name of person
109    HashMap<Integer, String> lastModifiedTime; // stores last modified time of person
```

```

-/Desktop/facdetec/FdActivity.java - Sublime Text (UNREGISTERED)
100    private int          mAbsoluteFaceSize = 0;                      // specifies absolute face size
101
102    private CameraBridgeViewBase mOpenCvCameraView;                    // for connecting opencv with camera
103    String id;                                         // store the roll id
104    ArrayList<Mat> sourceImages;           // stores the images
105    List<Integer> personIndexList;        // stores indices of person
106    List<Integer> namesIntList;          // stores roll number of person
107    HashMap<Integer, Integer> boundaryBoxColor; // stores box color of person
108    HashMap<Integer, String> personName;       // stores name of person
109    HashMap<Integer, String> lastModifiedTime; // stores last modified time of person
110    DBHelper myDb;                         // instance of student database
111    DBImgHelper imgDb;                     // instance of student image database
112    boolean isTrainComplete = false;         // specifies if training is complete
113
114    private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) { // this calls the opencv manager to load opencv modules
115        @Override
116        public void onManagerConnected(int status) { // this overrides the method declared in parent class of opencv
117            switch (status) { // this method handles the connection of camera with opencv
118                case LoaderCallbackInterface.SUCCESS: // based on status of connection with opencv manager, it chooses different paths
119                {
120                    // Log.i(TAG, "OpenCV loaded successfully");
121                    System.loadLibrary("detection_based_tracker"); // load the native c++ based tracker
122
123                    try { // load cascade file from application resources
124                        InputStream is = getResources().openRawResource(R.raw.haarcascade_frontalface_default);
125                        File cascadeDir = getDir("cascade", Context.MODE_PRIVATE); // create a directory named "cascade"
126
127                        mCascadeFile = new File(cascadeDir, "haarcascade_frontalface_default.xml"); // create a new file for storing haar_cascade classifier
128                        FileOutputStream os = new FileOutputStream(mCascadeFile); // for writing into the file
129
130                        byte[] buffer = new byte[4096]; // reading bytes from original haarcascade
131                        int bytesRead = 0; // number of bytes read from file
132                        while ((bytesRead = is.read(buffer)) != -1) { // if the file has not become empty yet
133                            os.write(buffer, 0, bytesRead); // write the contents into new file
134                        }
135                        is.close(); // close the main file
136                        os.close(); // close the new file
137
138                        // load the cascade classifier file for face detection
139                        mJavaDetector = new CascadeClassifier(mCascadeFile.getAbsolutePath());
140                        if (mJavaDetector.empty()) { // if there is no such file
141                            // Log.e(TAG, "Failed to load cascade classifier");
142                            mJavaDetector = null; // make this variable null
143                        }
144
145                        mNativeDetector = new DetectionBasedTracker(mCascadeFile.getAbsolutePath(), 0); // load the c++ based detection
146                        cascadeDir.delete(); // delete the cascade directory
147
148                    } catch (IOException e) { // if there is any sort of input/output error
149                        e.printStackTrace(); // print the error onto error stack
150                        // Log.e(TAG, "Failed to load cascade. Exception thrown: " + e);
151                    }
152
153                    mOpenCvCameraView.enableView(); // start the opencv camera
154                }

```

```

-/Desktop/facdetec/FdActivity.java - Sublime Text (UNREGISTERED)
154         }
155         break;
156     default: {
157         super.onManagerConnected(status); // for default cases
158     }
159     break;
160 }
161 }
162 };
163
164 FaceRecognizer faceRecognizer;
165 public FdActivity() {
166     mDetectorName = new String[2];
167     mDetectorName[JAVA_DETECTOR] = "Java";
168     mDetectorName[NATIVE_DETECTOR] = "Native (tracking)";
169 }
170 }
171
172 @Override
173 public void onCreate(Bundle savedInstanceState) {
174     // Log.i(TAG, "called onCreate");
175     super.onCreate(savedInstanceState);
176     id = getIntent().getExtras().getString("id");
177     getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
178     setContentView(R.layout.face_detect_surface_view);
179     // set the camera view
180     mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.fd_activity_surface_view);
181     mOpenCvCameraView.setVisibility(CameraBridgeViewBase.VISIBLE);
182     mOpenCvCameraView.setCvCameraViewListener(this);
183 }
184
185 @Override
186 public void onPause() {
187     super.onPause();
188     if (mOpenCvCameraView != null) {
189         mOpenCvCameraView.disableView();
190     }
191 }
192
193 @Override
194 public void onResume() {
195     super.onResume();
196     if (!OpenCVLoader.initDebug()) {
197         // Log.d(TAG, "Internal OpenCV library not found. Using OpenCV Manager for initialization");
198         // load the opencv java handler
199         OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_4_0, this, mLoaderCallback);
200     } else {
201         // load the opencv manager installed on the android phone
202         mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
203     }
204 }
205
206 public void onDestroy() {
207     super.onDestroy();
208     mOpenCvCameraView.disableView();
209 }

```

Line 225, Column 42

Tab Size: 4 Java

```

-/Desktop/facdetec/FdActivity.java - Sublime Text (UNREGISTERED)
202     mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
203 }
204 }
205
206 public void onDestroy() { // this method handles the tasks to be done when it exits
207     super.onDestroy(); // destroy the activity
208     mOpenCvCameraView.disableView(); // stop the camera
209 }
210
211 public void onCameraViewStarted(int width, int height) { // this function specifies the tasks to be done when camera starts
212     mGray = new Mat(); // create empty mat
213     mRgba = new Mat(); // create empty mat
214 }
215
216 public void onCameraViewStopped() { // this function specifies the tasks to be done when camera stops
217     mGray.release(); // clear the memory used
218     mRgba.release(); // clear the memory used
219 }
220
221 private Scalar getBoxColor(int boxColor) { // this method return the colour value corresponding to box color value
222     if(boxColor < 5) { // if less than 5, return red
223         return new Scalar(255, 0, 0);
224     } else if(boxColor < 8) { // if between 5 and 7, return blue
225         return new Scalar(0, 0, 255);
226     } else { // return green
227         return new Scalar(0, 255, 0);
228     }
229 }
230
231 private String getTime() { // for getting time
232     // extract time in format as specified
233     SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss", Locale.getDefault());
234     Date date = new Date(); // initialise a date variable
235     String currentTime = dateFormat.format(date); // extract the time
236     Log.d(TAG, nd);
237     return currentTime; // return the time
238 }
239
240 private long getTimeDiff(String globalTime, String playerTime) { // for getting the time difference
241     // extract time in format as specified
242     DateFormat df = new SimpleDateFormat("HH:mm:ss", Locale.getDefault());
243     try {
244         Date date1 = df.parse(globalTime); // convert string to time
245         Date date2 = df.parse(playerTime); // convert string to time
246         return date1.getTime() - date2.getTime(); // return their difference
247     } catch (ParseException e) { // if there is any exception
248         Log.d(TAG, "error in timediff");
249         return 0; // return 0
250     }
251 }
252
253 public void training() { // this method handles the training of face recogniser
254     myDb = new DBHelper(this); // instance of the student record database
255     imDb = new DBImgHelper(this); // instance of the student image database
256

```

Line 225, Column 42 | Tab Size: 4 | Java

```

-/Desktop/facdetec/FdActivity.java - Sublime Text (UNREGISTERED)
253     public void training() {
254         myDb = new DBHelper(this);
255         imDb = new DBImgHelper(this);
256     }
257
258     Cursor result = myDb.getCourseData(id);
259     Mat imgs = new Mat();
260     if(result.getCount() == 0) {
261         finish();
262     } else {
263         sourceImages = new ArrayList<>();
264         personIndexList = new ArrayList<>();
265         namesIntList = new ArrayList<>();
266         boundaryBoxColor = new HashMap<>();
267         personName = new HashMap<>();
268         lastModifiedTime = new HashMap<>();
269
270         String name, modifiedTime;
271         int id, boxColor;
272         result.moveToFirst();
273         do{
274             name = result.getString(1);
275             id = result.getInt(0);
276             modifiedTime = result.getString(4);
277             boxColor = result.getInt(5);
278             ArrayList<Bitmap> allImages = imDb.getImg(id);
279             personIndexList.add(id);
280             personName.put(id, name);
281             boundaryBoxColor.put(id, boxColor);
282             lastModifiedTime.put(id, modifiedTime);
283
284             for (Bitmap img : allImages) {
285                 Mat mat = new Mat();
286                 Bitmap copying = img.copy(Bitmap.Config.RGB_565, true);
287                 Utils.bitmapToMat(copying, mat);
288                 Imgproc.cvtColor(mat, mat, Imgproc.COLOR_BGR2GRAY);
289                 imgs = mat;
290                 MatOfRect faces = new MatOfRect();
291                 Imgproc.detectMultiScale(mat, faces, 1.1, 2, new Size(mAbsoluteFaceSize, mAbsoluteFaceSize), new Size());
292
293                 Rect[] facesArray = faces.toArray();
294                 if(facesArray.length > 0) {
295                     Rect roi = new Rect(facesArray[0].tl(), facesArray[0].br());
296                     Mat cropped = new Mat(mat, roi);
297                     namesIntList.add(personIndexList.indexOf(id));
298                     sourceImages.add(cropped);
299                 }
300             }
301         }while (result.moveToNext());
302         faceRecognizer = LBPHFaceRecognizer.create(); // instantiate the lbph recognizer
303     }
304 }

```

```

~/Desktop/facetedet/FdActivity.java - Sublime Text (UNREGISTERED)
304         }
305     }
306     }while (result.moveToFirst());
307     faceRecognizer = LBPHFaceRecognizer.create();           // instantiate the lbph recogniser
308     MatOfInt matOfInt = new MatOfInt();                   // create empty mat of int
309     matOfInt.fromList(namesIntList);                      // convert list to mat of int
310     faceRecognizer.train(sourceImages, matOfInt);        // train the face recogniser
311     isTrainComplete = true;                             // make isTrainComplete to true
312   }
313 }
314
315 @Override
316 public void onBackPressed() {                         // this functions overrides the function defined in the parent class "AppCompatActivity".
317   for(Integer rollNumber : personIndexList) {          // this method handles the updation of database when back is pressed
318     updateModifiedTime(rollNumber.longValue(), boundaryBoxColor.get(rollNumber), lastModifiedTime.get(rollNumber)); // iterating over all persons
319     boolean isDone = myDb.updateModifiedTime(rollNumber.longValue(), boundaryBoxColor.get(rollNumber), lastModifiedTime.get(rollNumber));
320     if(!isDone) {                                     // if updation is not done
321       // show "not updated" onto screen
322       Toast.makeText(getApplicationContext(), "Not Updated", Toast.LENGTH_LONG).show();
323     }
324   }
325   finish();                                         // finish the activity
326 }
327
328 public Mat onCameraFrame(CvCameraViewFrame inputFrame) { // this method handles the recognition part
329   mRgba = inputFrame.rgba();                          // extract the rgba form of input frame
330   mGray = inputFrame.gray();                         // extract the gray form of input frame
331   if(!isTrainComplete) {                            // if the training is not complete, then train the dataset
332     training();
333   }
334   String globalTime = getTime();                     // get the time of running this frame
335   if (mAbsoluteFaceSize == 0) {                      // if absolute face size set to 0
336     int height = mGray.rows();                      // get the height of frame
337     if (Math.round(height * mRelativeFaceSize) > 0) { // if new face size is not 0, then make it new face size
338       mAbsoluteFaceSize = Math.round(height * mRelativeFaceSize);
339     }
340     mNativeDetector.setMinFaceSize(mAbsoluteFaceSize); // set the face size for native detector
341   }
342
343   MatOfRect faces = new MatOfRect();                 // creating empty mat of rect
344
345   if (mJavaDetector != null) {                        // if there is java detector
346     mJavaDetector.detectMultiScale(mGray, faces, 1.1, 2, 2,
347     new Size(mAbsoluteFaceSize, mAbsoluteFaceSize), new Size()); // set the detector with face size
348   } else {                                           // if not
349     finish();                                         // then finish the activity
350   }
351
352   Rect[] facesArray = faces.toArray();               // stores the corner of detected faces
353   for (int i = 0; i < facesArray.length; i++) {      // iterating over all faces detected
354     Rect roi = new Rect(facesArray[i].tL(), facesArray[i].br()); // create empty rectangle
355     Mat cropped = new Mat(mGray, roi);                // crop the face
356     int[] label = new int[1];                         // used for storing label of identified person
357     double[] conf = new double[1];                    // confidence value
358     try {                                            // predict the face
359       ...
360     } catch (Exception e) {
361       Log.e("Error", "Error in face detection");
362     }
363   }

```

Line 256, Column 1 | Tab Size: 4 | Java

```

-/Desktop/facdetec/FdActivity.java - Sublime Text (UNREGISTERED)
355     Mat cropped = new Mat(mGray, roi);                                // crop the face
356     int[] label = new int[1];                                         // used for storing label of identified person
357     double[] conf = new double[1];                                     // confidence value
358     try {
359         faceRecognizer.predict(cropped, label, conf);                  // predict the face
360     } catch (Exception e) {
361         Log.d(TAG, "Nothing");
362     }
363     Random randomVal = new Random();                                    // used for generating random values
364
365     if(label[0] < 0 || conf[0] > 150.0) {                                // criteria for unknown face
366         Imgproc.rectangle(mRgba, facesArray[i].tl(), facesArray[i].br(), new Scalar(255,255,255), 2);
367         Imgproc.putText(mRgba, "Unknown "+conf[0], new Point(x,y), Core.FONT_HERSHEY_PLAIN, 1.0, new Scalar(255,255,255));
368     } else {
369         int roll = personIndexList.get(label[0]);                         // get the roll number of the student
370         String playerTime = lastModifiedTime.get(roll);                  // get time of checking
371         int boxColor = boundaryBoxColor.get(roll);                        // get the boxcolor
372         if(getTimeDiff(globalTime, playerTime) > 60000) {                // if the time differnce is greater than 1 minute, change the boxcolor and modification time
373             int newValue = randomVal.nextInt(9) + 1;
374             boxColor = newValue;
375             lastModifiedTime.put(roll, globalTime);
376             boundaryBoxColor.put(roll, newValue);
377         }
378         Imgproc.rectangle(mRgba, facesArray[i].tl(), facesArray[i].br(), getBoxColor(boxColor), 2);
379         Imgproc.putText(mRgba, personName.get(roll)+" "+conf[0], new Point(x,y), Core.FONT_HERSHEY_PLAIN, 1.0, getBoxColor(boxColor));
380     }
381 }
382
383     return mRgba;                                                 // return the final camera frame
384 }
385
386 @Override
387 public boolean onCreateOptionsMenu(Menu menu) {                      // this functions overrides the function defined in the parent class "Menu".
388     // Log.i(TAG, "called onCreateOptionsMenu");
389     getMenuInflater().inflate(R.menu.main_menu, menu);                   // this method handles the creation of menu for face size
390
391     mItemFace50 = menu.add("Face size 50%");                         // add the menu item with size 50%
392     mItemFace40 = menu.add("Face size 40%");                         // add the menu item with size 40%
393     mItemFace30 = menu.add("Face size 30%");                         // add the menu item with size 30%
394     mItemFace20 = menu.add("Face size 20%");                         // add the menu item with size 20%
395     mItemType = menu.add(mDetectorName[mDetectorType]);               // add the menu item for detector type
396
397     return true;                                                       // return true when complete
398 }
399
400 @Override
401 public boolean onOptionsItemSelected(MenuItem item) {              // this functions overrides the function defined in the parent class "Menu".
402     // this method handles the work to be done when a particular option is selected
403
404     if (item == mItemFace50) {                                         // if face size 50% is seleted, set face size to 50%
405         setMinFaceSize(0.5f);
406     } else if (item == mItemFace40) {                                     // if face size 40% is seleted, set face size to 40%
407         setMinFaceSize(0.4f);
408     } else if (item == mItemFace30) {                                     // if face size 30% is seleted, set face size to 30%
409         setMinFaceSize(0.3f);
410     } else if (item == mItemFace20) {                                     // if face size 20% is seleted, set face size to 20%
411         setMinFaceSize(0.2f);
412     } else if (item == mItemType) {                                       // if detection type is selected then set the detector as selected
413         int tmpDetectorType = (mDetectorType + 1) % mDetectorName.length;
414         item.setTitle(mDetectorName[tmpDetectorType]);                  // set the new menu option title

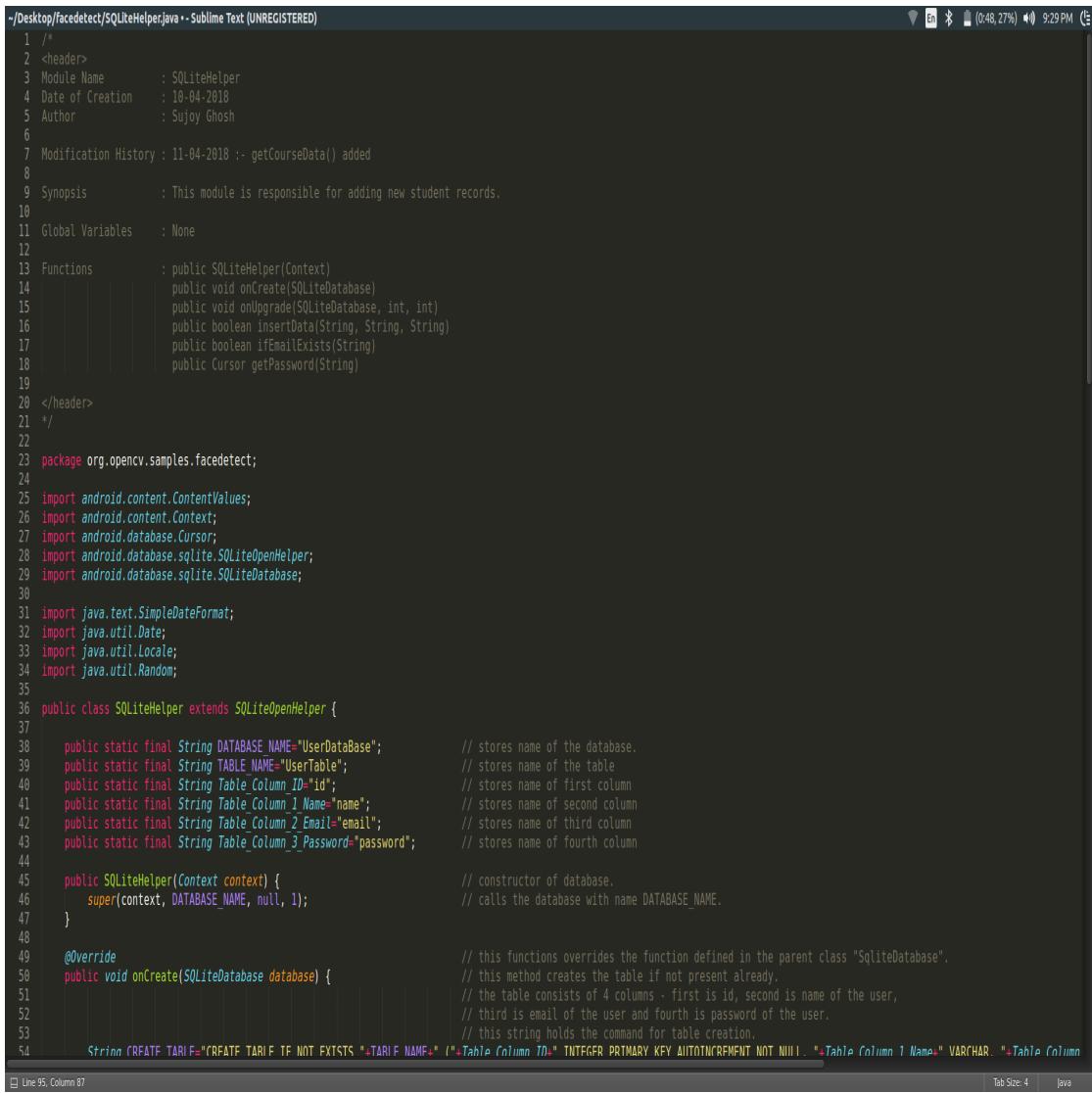
```

Line 256, Column 1

Tab Size: 4 Java

```
/Desktop/facetedet/FdActivity.java - Sublime Text (UNREGISTERED)
388     mItemFace50 = menu.add("Face size 50%");           // add the menu item with size 50%
389     mItemFace40 = menu.add("Face size 40%");           // add the menu item with size 40%
390     mItemFace30 = menu.add("Face size 30%");           // add the menu item with size 30%
391     mItemFace20 = menu.add("Face size 20%");           // add the menu item with size 20%
392     mItemType = menu.add(mDetectorName[mDetectorType]); // add the menu item for detector type
393     return true;                                      // return true when complete
394 }
395
396 @Override
397 public boolean onOptionsItemSelected(MenuItem item) { // this functions overrides the function defined in the parent class "Menu".
398     if (item == mItemFace50) {                         // this method handles the work to be done when a particular option is selected
399         setMinFaceSize(0.5f);                           // if face size 50% is selected, set face size to 50%
400     } else if (item == mItemFace40) {
401         setMinFaceSize(0.4f);                           // if face size 40% is selected, set face size to 40%
402     } else if (item == mItemFace30) {
403         setMinFaceSize(0.3f);                           // if face size 30% is selected, set face size to 30%
404     } else if (item == mItemFace20) {
405         setMinFaceSize(0.2f);                           // if face size 20% is selected, set face size to 20%
406     } else if (item == mItemType) {                     // if detection type is selected then set the detector as selected
407         int tmpDetectorType = (mDetectorType + 1) % mDetectorName.length;
408         item.setTitle(mDetectorName[tmpDetectorType]); // set the new menu option title
409         setDetectorType(tmpDetectorType);              // set detector type
410     }
411     return true;
412 }
413
414 private void setMinFaceSize(float faceSize) {        // this method is used to set face size for detection
415     mRelativeFaceSize = faceSize;                      // set relative face size
416     mAbsoluteFaceSize = 0;                            // set absolute face size to 0
417 }
418
419
420 private void setDetectorType(int type) {             // this method handles native detector start/stop
421     if (mDetectorType != type) {                      // if type is not set, then set it
422         mDetectorType = type;                         // set the detector type to as given detector type
423
424         if (type == NATIVE_DETECTOR) {                // if type is native detector
425             mNativeDetector.start();                  // start the native detector
426         } else {                                     // if not of native detector, then stop it.
427             mNativeDetector.stop();
428         }
429     }
430 }
431 }
432 }
```

# 10 User Database



The screenshot shows a Sublime Text editor window with the following details:

- File Path: ~/Desktop/facedetect/SQLiteHelper.java
- Text: Sublime Text (UNREGISTERED)
- System Status Bar: En, 0:48, 27%, 9:29 PM, battery icon
- Code Content:

```
1  /*
2   <header>
3   Module Name      : SQLiteHelper
4   Date of Creation : 10-04-2018
5   Author          : Sujoy Ghosh
6
7   Modification History : 11-04-2018 :- getCourseData() added
8
9   Synopsis        : This module is responsible for adding new student records.
10  Global Variables : None
11
12  Functions       : public SQLiteHelper(Context)
13          |         public void onCreate(SQLiteDatabase)
14          |         public void onUpgrade(SQLiteDatabase, int, int)
15          |         public boolean insertData(String, String, String)
16          |         public boolean ifEmailExists(String)
17          |         public Cursor getPassword(String)
18
19  </header>
20 */
21
22 package org.opencv.samples.facedetect;
23
24
25 import android.content.ContentValues;
26 import android.content.Context;
27 import android.database.Cursor;
28 import android.database.sqlite.SQLiteOpenHelper;
29 import android.database.sqlite.SQLiteDatabase;
30
31 import java.text.SimpleDateFormat;
32 import java.util.Date;
33 import java.util.Locale;
34 import java.util.Random;
35
36 public class SQLiteHelper extends SQLiteOpenHelper {
37
38     public static final String DATABASE_NAME="UserDataBase";
39     public static final String TABLE_NAME="UserTable";
40     public static final String Table_Column_ID="id";
41     public static final String Table_Column_1_Name="name";
42     public static final String Table_Column_2_Email="email";
43     public static final String Table_Column_3_Password="password";
44
45     public SQLiteHelper(Context context) {
46         super(context, DATABASE_NAME, null, 1);
47     }
48
49     @Override
50     public void onCreate(SQLiteDatabase database) {
51
52         String CREATE_TABLE="CREATE TABLE IF NOT EXISTS "+TABLE_NAME+" ("+Table_Column_ID+" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL , "+Table_Column_1_Name+" VARCHAR , "+Table_Column_2_Email+" VARCHAR , "+Table_Column_3_Password+" VARCHAR )";
53
54     }
55 }
```
- Bottom Status Bar: Line 95, Column 87, Tab Size: 4, Java

```

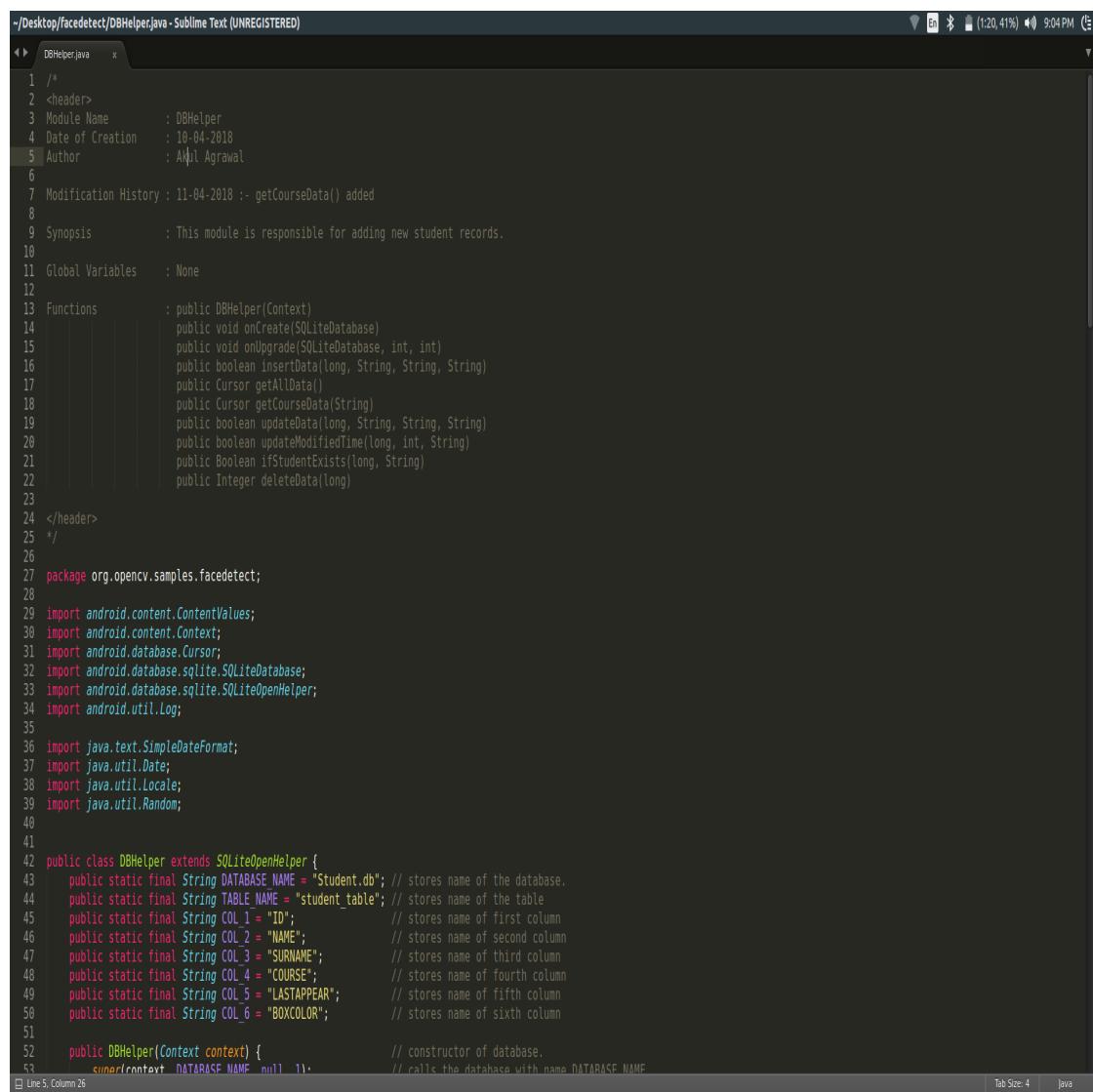
-/Desktop/facdetec/SQLiteHelper.java - Sublime Text (UNREGISTERED)
46     super(context, DATABASE_NAME, null, 1);           // calls the database with name DATABASE_NAME.
47 }
48
49 @Override
50 public void onCreate(SQLiteDatabase database) {
51
52     String CREATE_TABLE="CREATE TABLE IF NOT EXISTS "+TABLE_NAME+" ("+Table_Column_ID+" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "+Table_Column_1_Name+" VARCHAR, "+Table_Column_2_
53     database.execSQL(CREATE_TABLE);                  // execute the string query.
54 }
55
56
57 @Override
58 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { // this method handles the upgradation of database.
59     db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);          // this deletes the previously existing table,
60     onCreate(db);                                         // and creates a new table.
61 }
62
63
64 public boolean insertData(String name,String email,String password) { // this method handles the insertion of new row into database.
65     SQLiteDatabase db = this.getWritableDatabase();           // Open the database for reading.
66     ContentValues contentValues = new ContentValues();    // declare variable to store the new row to be inserted into database.
67     contentValues.put(Table_Column_1_Name, name);          // put name into it.
68     contentValues.put(Table_Column_2_Email, email);        // put email into it.
69     contentValues.put(Table_Column_3_Password, password); // put password into it.
70     long result = db.insert(TABLE_NAME,null ,contentValues); // insert row into database.
71     if(result == -1) {                                // if insertion is incomplete, return false.
72         return false;                                // else return true, denoting insertion is complete.
73     } else {
74         return true;
75     }
76 }
77
78 public boolean ifEmailExists(String inputEmail) { // this method checks whether the entered email already exists or not.
79     SQLiteDatabase db = this.getWritableDatabase();          // Open the database for reading.
80
81     Cursor res = db.rawQuery("select * from "+TABLE_NAME+" where email='"+inputEmail+"'"+null); // execute the query to check for the entered email address.
82     int count = res.getCount();                         // count the number of entries having email equal to given email.
83     if(count == 0) {                                  // if count value is 0, so there is no email of given input, return false
84     {
85         return false;
86     } else{                                         // if count is positive, then ther is email as given and return true, denoting email
87         return true;                                // is present.
88     }
89 }
90
91 public Cursor getPassword(String email) { // this method handles the fetching of password
92     SQLiteDatabase db = this.getWritableDatabase();          // Open the database for reading.
93
94     Cursor res = db.rawQuery("select * from "+TABLE_NAME+" where email='"+email+"'"+null); // fetch the password corresponding to given email id
95     return res;                                     // return the password cursor.
96 }
97
98 }
99

```

Line 73, Column 81, Saved ~[Desktop]/facdetec/SQLiteHelper.java (UTF-8)

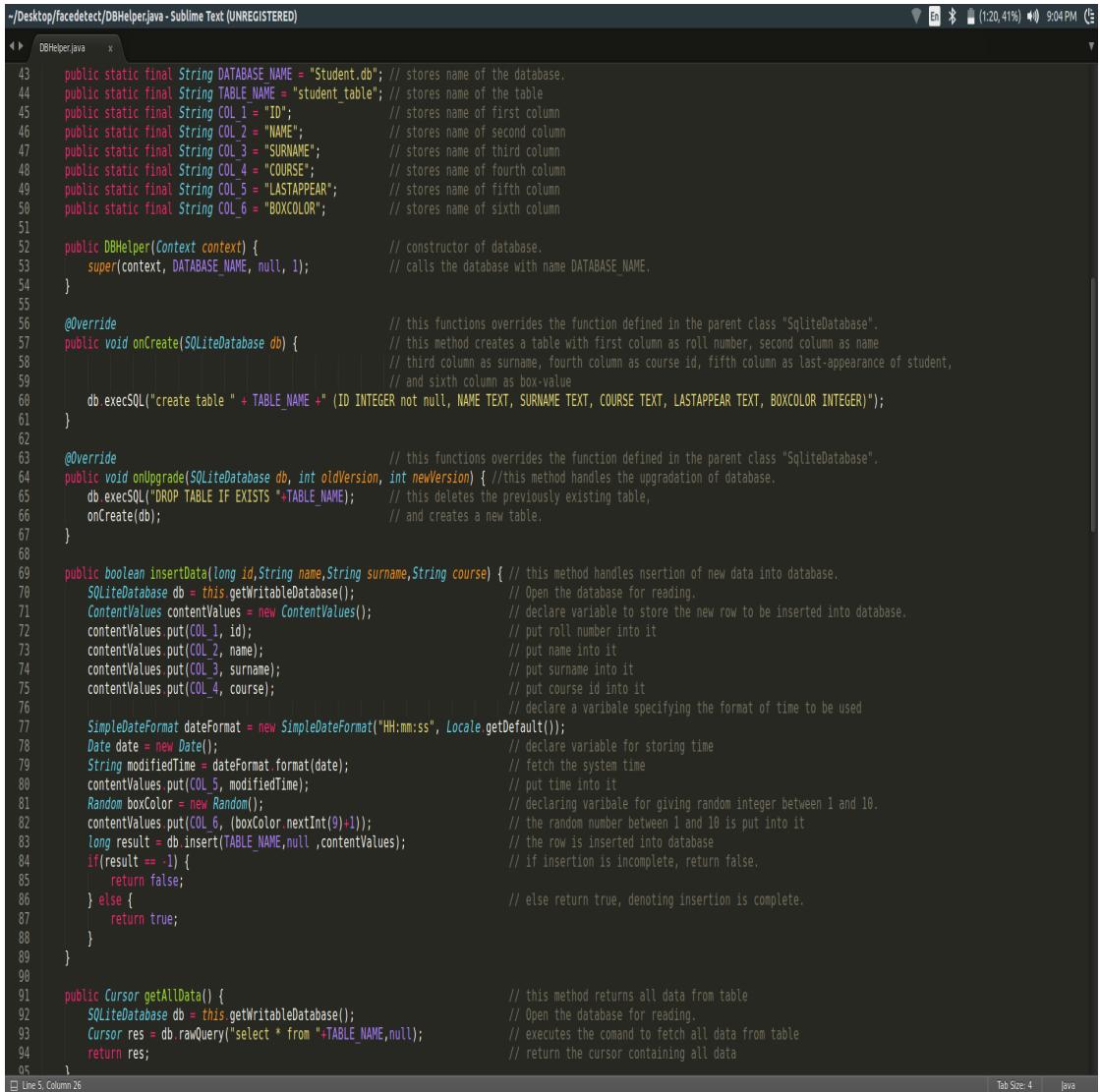
Tab Size: 4 Java

# 11 Student Database



The screenshot shows a Sublime Text window with the file 'DBHelper.java' open. The code is a Java class for managing a SQLite database. It includes comments for module details, modification history, synopsis, global variables, and function descriptions. The class extends SQLiteOpenHelper and defines static final variables for database name, table name, and column names. It also includes a constructor and several database manipulation methods.

```
/*  
<header>  
Module Name : DBHelper  
Date of Creation : 10-04-2018  
Author : Ajit Agrawal  
Modification History : 11-04-2018 :- getCourseData() added  
Synopsis : This module is responsible for adding new student records.  
Global Variables : None  
Functions : public DBHelper(Context)  
           public void onCreate(SQLiteDatabase)  
           public void onUpgrade(SQLiteDatabase, int, int)  
           public boolean insertData(long, String, String, String)  
           public Cursor getAllData()  
           public Cursor getCourseData(String)  
           public boolean updateData(long, String, String, String)  
           public boolean updateModifiedTime(long, int, String)  
           public Boolean ifstudentExists(long, String)  
           public Integer deleteData(long)  
</header>  
*/  
  
package org.opencv.samples.facedetect;  
  
import android.content.ContentValues;  
import android.content.Context;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
import android.util.Log;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.Locale;  
import java.util.Random;  
  
public class DBHelper extends SQLiteOpenHelper {  
    public static final String DATABASE_NAME = "Student.db"; // stores name of the database.  
    public static final String TABLE_NAME = "student_table"; // stores name of the table  
    public static final String COL_1 = "ID"; // stores name of first column  
    public static final String COL_2 = "NAME"; // stores name of second column  
    public static final String COL_3 = "SURNAME"; // stores name of third column  
    public static final String COL_4 = "COURSE"; // stores name of fourth column  
    public static final String COL_5 = "LASTAPPEAR"; // stores name of fifth column  
    public static final String COL_6 = "BOXCOLOR"; // stores name of sixth column  
  
    public DBHelper(Context context) { // constructor of database.  
        super(context, DATABASE_NAME, null, 1); // calls the database with name DATABASE_NAME  
    }  
}
```



The screenshot shows a Sublime Text window with the file 'DBHelper.java' open. The code is a Java class for managing a SQLite database. It defines constants for database and table names, and columns. It overrides the onCreate and onUpgrade methods to create the table and handle upgrades. It also implements a method to insert data into the table, handling time and random box colors. Finally, it provides a method to get all data from the table.

```

1  public static final String DATABASE_NAME = "Student.db"; // stores name of the database.
2  public static final String TABLE_NAME = "student_table"; // stores name of the table
3  public static final String COL_1 = "ID"; // stores name of first column
4  public static final String COL_2 = "NAME"; // stores name of second column
5  public static final String COL_3 = "SURNAME"; // stores name of third column
6  public static final String COL_4 = "COURSE"; // stores name of fourth column
7  public static final String COL_5 = "LASTAPPEAR"; // stores name of fifth column
8  public static final String COL_6 = "BOXCOLOR"; // stores name of sixth column
9
10 public DBHelper(Context context) { // constructor of database.
11     super(context, DATABASE_NAME, null, 1); // calls the database with name DATABASE_NAME.
12 }
13
14 @Override // this functions overrides the function defined in the parent class "SqliteDatabase".
15 public void onCreate(SQLiteDatabase db) { // this method creates a table with first column as roll number, second column as name
16     // third column as surname, fourth column as course id, fifth column as last-appearance of student,
17     // and sixth column as box-value
18     db.execSQL("create table " + TABLE_NAME + " (ID INTEGER not null, NAME TEXT, SURNAME TEXT, COURSE TEXT, LASTAPPEAR TEXT, BOXCOLOR INTEGER)");
19 }
20
21 @Override // this functions overrides the function defined in the parent class "SqliteDatabase".
22 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { // this method handles the upgradation of database.
23     db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME); // this deletes the previously existing table,
24     onCreate(db); // and creates a new table.
25 }
26
27 public boolean insertData(long id, String name, String surname, String course) { // this method handles insertion of new data into database.
28     SQLiteDatabase db = this.getWritableDatabase(); // Open the database for reading.
29     ContentValues contentValues = new ContentValues(); // declare variable to store the new row to be inserted into database.
30     contentValues.put(COL_1, id); // put roll number into it
31     contentValues.put(COL_2, name); // put name into it
32     contentValues.put(COL_3, surname); // put surname into it
33     contentValues.put(COL_4, course); // put course Id into it
34     contentValues.put(COL_5, DateFormat.format("HH:mm:ss", Locale.getDefault())); // declare a varibale specifying the format of time to be used
35     Date date = new Date(); // declare variable for storing time
36     String modifiedTime = dateFormat.format(date); // fetch the system time
37     contentValues.put(COL_5, modifiedTime); // put time into it
38     Random boxColor = new Random(); // declaring varibale for giving random integer between 1 and 10.
39     contentValues.put(COL_6, (boxColor.nextInt(9)+1)); // the random number between 1 and 10 is put into it
40     long result = db.insert(TABLE_NAME, null, contentValues); // the row is inserted into database
41     if(result == -1) { // if insertion is incomplete, return false.
42         return false;
43     } else { // else return true, denoting insertion is complete.
44         return true;
45     }
46 }
47
48 public Cursor getAllData() { // this method returns all data from table
49     SQLiteDatabase db = this.getWritableDatabase(); // Open the database for reading.
50     Cursor res = db.rawQuery("select * from "+TABLE_NAME,null); // executes the command to fetch all data from table
51     return res; // return the cursor containing all data
52 }

```

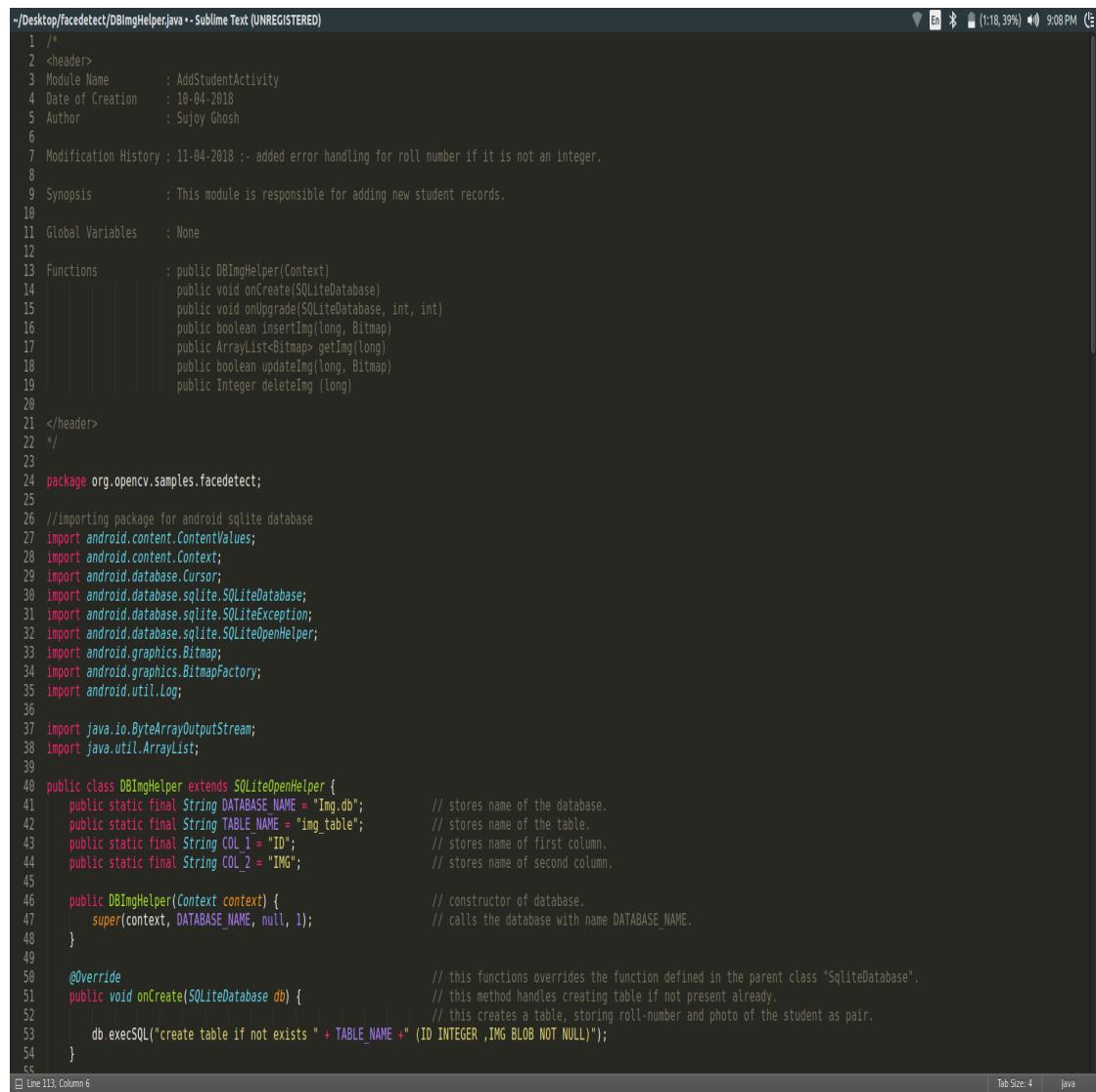
```

File Edit Selection Find View Goto Tools Project Preferences Help
97     public Cursor getCourseData(String CourseID) {           // this method returns all data where course id is same as input parameter
98         SQLiteDatabase db = this.getWritableDatabase();          // Open the database for reading.
99         Cursor res = db.rawQuery("select * from "+TABLE_NAME+" where COURSE='"+CourseID+"'"+",null);    // executes the command to fetch all data from table where course id is same as input parameter
100        return res;                                         // return the cursor containing the required data
101    }
102
103
104    public Boolean ifStudentExists(long id, String courseId) { // this method handles the presence of student in particular course
105        SQLiteDatabase db = this.getWritableDatabase();          // Open the database for reading,
106        Cursor res = db.rawQuery("select * from "+TABLE_NAME+" where COURSE='"+courseId+"'"+",null); // get list of all students in input course
107        if(res.getCount()>0) {                                // if the course exists in database
108            res.moveToFirst();                                // iterate over all students
109            do {
110                String rollID = res.getString(0);               // extract the roll number of the student
111                if (rollID.equals(String.valueOf(id))) {        // if they are equal, return true
112                    return Boolean.TRUE;
113                }
114            } while (res.moveToNext());
115        }
116        return Boolean.FALSE;                                // if no student roll number matches, return false.
117    }
118
119
120    public boolean updateData(long id, String name, String surname, String course) { // handles updating of data in table
121        SQLiteDatabase db = this.getWritableDatabase();          // Open the database for reading.
122        ContentValues contentValues = new ContentValues();      // declare variable to store the new row to be inserted into database.
123        contentValues.put(COL_2, name);                         // put name into it
124        contentValues.put(COL_3, surname);                      // put surname into it
125        contentValues.put(COL_4, course);                      // put course id into it
126        contentValues.put(COL_5, dateFormat.format(date));       // declare a variable specifying the format of time to be used
127        SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss", Locale.getDefault());
128        Date date = new Date();                               // declare variable for storing time
129        String modifiedTime = dateFormat.format(date);        // fetch the system time
130        contentValues.put(COL_5, modifiedTime);                // put last-appear time into it
131        Random boxColor = new Random();                     // variable for generating random numbers
132        contentValues.put(COL_6, (boxColor.nextInt(9)+1));     // generate random number between 1 and 10 and put boxcolor into it
133        db.update(TABLE_NAME, contentValues, "ID = "+id,null ); // update the table for roll number as given with new values as stored
134        return true;                                         // return true denoting update completion
135    }
136
137
138    public boolean updateModifiedTime(long id, int boxColor, String modifiedTime){ // this method is used to modify the boxcolor and last-appear time for given roll id
139        SQLiteDatabase db = this.getWritableDatabase();          // Open the database for reading.
140        ContentValues contentValues = new ContentValues();      // declare variable to store the new row to be inserted into database.
141        contentValues.put(COL_5, modifiedTime);                 // put last-appear time into it
142        contentValues.put(COL_6, boxColor);                   // put boxcolor into it
143        db.update(TABLE_NAME, contentValues, "ID = "+id,null ); // update the table for roll number as given with new values as stored
144        return true;                                         // return true denoting update completion
145    }
146
147    public Integer deleteData (long id) {                  // this method is used to delete data corresponding to given roll number
148        SQLiteDatabase db = this.getWritableDatabase();          // Open the database for reading.
149        return db.delete(TABLE_NAME, "ID = "+id,null);        // return the number of rows deleted, if roll number is not present, it returns 0, else returns positive value
150    }
151

```

Line 5, Column 26 | Tab Size: 4 | Java

# 12 Student Image Database



The screenshot shows a Sublime Text editor window with the following details:

- File Path: ~/Desktop/facdetect/DBImgHelper.java
- Text: UNREGISTERED
- Status Bar: En (1:18, 39%) 9:08 PM
- Code Content:

```
1 /*
2 <header>
3 Module Name      : AddstudentActivity
4 Date of Creation : 10-04-2018
5 Author           : Sujoy Ghosh
6
7 Modification History : 11-04-2018 :- added error handling for roll number if it is not an integer.
8
9 Synopsis         : This module is responsible for adding new student records.
10
11 Global Variables : None
12
13 Functions        : public DBImgHelper(Context)
14          |     public void onCreate(SQLiteDatabase)
15          |     public void onUpgrade(SQLiteDatabase, int, int)
16          |     public boolean insertImg(long, Bitmap)
17          |     public ArrayList<Bitmap> getImg(long)
18          |     public boolean updateImg(long, Bitmap)
19          |     public Integer deleteImg (long)
20
21 </header>
22 */
23
24 package org.opencv.samples.facdetect;
25
26 //importing package for android sqlite database
27 import android.content.ContentValues;
28 import android.content.Context;
29 import android.database.Cursor;
30 import android.database.sqlite.SQLiteDatabase;
31 import android.database.sqlite.SQLiteException;
32 import android.database.sqlite.SQLiteOpenHelper;
33 import android.graphics.Bitmap;
34 import android.graphics.BitmapFactory;
35 import android.util.Log;
36
37 import java.io.ByteArrayOutputStream;
38 import java.util.ArrayList;
39
40 public class DBImgHelper extends SQLiteOpenHelper {
41     public static final String DATABASE_NAME = "1mg.db";
42     public static final String TABLE_NAME = "img_table";
43     public static final String COL_1 = "ID";
44     public static final String COL_2 = "IMG";
45
46     public DBImgHelper(Context context) {
47         super(context, DATABASE_NAME, null, 1);
48     }
49
50     @Override
51     public void onCreate(SQLiteDatabase db) {
52         db.execSQL("create table if not exists " + TABLE_NAME + " (" + COL_1 + " INTEGER ,IMG BLOB NOT NULL)");
53     }
54 }
```
- Bottom Status: Line 113, Column 6 | Tab Size: 4 | Java

```

~/Desktop/facdetec/DBImgHelper.java -- Sublime Text (UNREGISTERED)
46  public DBImgHelper(Context context) {           // constructor of database.
47      super(context, DATABASE_NAME, null, 1);       // calls the database with name DATABASE_NAME.
48  }
49
50  @Override                                     // this functions overrides the function defined in the parent class "SqliteDatabase".
51  public void onCreate(SQLiteDatabase db) {        // this method handles creating table if not present already.
52      db.execSQL("create table if not exists " + TABLE_NAME + " (ID INTEGER ,IMG BLOB NOT NULL)");
53  }
54
55  @Override                                     // this functions overrides the function defined in the parent class "SqliteDatabase".
56  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { // this method handles the upgradation of database
57      db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);
58      onCreate(db);
59  }
60
61  public boolean insertImg(long id, Bitmap bm) {   // this method handles insertion of new image of a student, returning true if inserted.
62      ByteArrayOutputStream out = new ByteArrayOutputStream(); // variable is declared to store compressed version of incoming image.
63      bm.compress(Bitmap.CompressFormat.PNG, 100, out); // the incoming image is compressed to PNG format and stored in 'out' variable.
64      byte[] buffer=out.toByteArray(); // Convert the compressed image into byte array for storing in database.
65      SQLiteDatabase db = this.getWritableDatabase(); // Open the database for writing.
66      db.beginTransaction(); // Start the transaction.
67
68      try {
69          ContentValues values = new ContentValues();
70          values.put("ID", id);
71          values.put("IMG", buffer);
72          long result=db.insert(TABLE_NAME, null, values);
73          db.setTransactionSuccessful();
74          if(result == -1) {
75              return false;
76          } else {
77              return true;
78          }
79      } catch (SQLiteException e) {
80          e.printStackTrace();
81          return false;
82      } finally {
83          db.endTransaction();
84          db.close();
85      }
86  }
87
88  public ArrayList<Bitmap> getImg(long id) {      // this method returns all images of given roll number.
89      ArrayList<Bitmap> allImages = new ArrayList<Bitmap>(); // this varisble stores all images of given roll number.
90      SQLiteDatabase db = this.getReadableDatabase(); // Open the database for reading.
91      db.beginTransaction(); // Start the transaction.
92
93      try {
94          String selectQuery = "SELECT * FROM "+ TABLE_NAME+" WHERE id = " +id; // query string to fetch all images.
95          Cursor cursor = db.rawQuery(selectQuery, null); // executing the sql query.
96          if(cursor.getCount() >0) { // if there are positive number of images.
97              cursor.moveToFirst(); // move the cursor head to first row.
98              while (cursor.moveToNext()) { // while cursor head reaches end position.
99                  Bitmap bitmap = null; // this variable temporarily stores the current image.
100                 byte[] blob = cursor.getBlob(cursor.getColumnIndex("IMG")); // extract the image from database cursor.
101                 bitmap=BitmapFactory.decodeByteArray(blob, 0, blob.length); // decode the image back to its original form

```

```

~/Desktop/facetedetector/DBImgHelper.java -- Sublime Text (UNREGISTERED) 9:09 PM ⓘ
79         } catch (SQLiteException e) { // if there is exception/error in inserting, print error.
80             e.printStackTrace(); // return false, denoting unsuccessful insertion
81         }
82     } finally { // this statement is always executed even if there was insertion failure.
83         db.endTransaction(); // end the transaction and save the changes.
84         db.close(); // close the database.
85     }
86 }
87
88 public ArrayList<Bitmap> getImg(long id) { // this method returns all images of given roll number.
89     ArrayList<Bitmap> allImages = new ArrayList<Bitmap>(); // this variable stores all images of given roll number.
90     SQLiteDatabase db = this.getReadableDatabase(); // Open the database for reading.
91     db.beginTransaction(); // Start the transaction.
92     try {
93         String selectQuery = "SELECT * FROM "+ TABLE_NAME+" WHERE id = " +id; // query string to fetch all images.
94         Cursor cursor = db.rawQuery(selectQuery, null); // executing the sql query.
95         if(cursor.getCount() >0) { // if there are positive number of images.
96             cursor.moveToFirst(); // move the cursor head to first row.
97             while (cursor.moveToNext()) { // while cursor head reaches end position.
98                 Bitmap bitmap = null; // this variable temporarily stores the current image.
99                 byte[] blob = cursor.getBlob(cursor.getColumnIndex("IMG")); // extract the image from database cursor.
100                bitmap= BitmapFactory.decodeByteArray(blob, 0, blob.length); // decode the image back to its original form.
101                allImages.add(bitmap); // add the image to the list of images.
102            }
103        }
104        db.setTransactionSuccessful(); // denote successful transaction has occurred.
105    } catch (SQLiteException e) { // if there is exception/error in fetching, print error.
106        e.printStackTrace();
107    }finally { // this statement is always executed even if there was insertion failure.
108        db.endTransaction(); // end the transaction and save the changes.
109        db.close(); // close the database.
110    }
111
112     return allImages;
113 }
114
115 public Integer deleteImg (long id) { // this function handles the deletion of image of given roll number.
116     SQLiteDatabase db = this.getWritableDatabase(); // Open the database for writing.
117     return db.delete(TABLE_NAME, "ID = " +id,null); // return the value of delete(), if there is any image present associated with roll number,
118     // then positive value is returned, else 0 is returned.(So, 0 means that roll number is not present)
119 }
120 }
121

```