

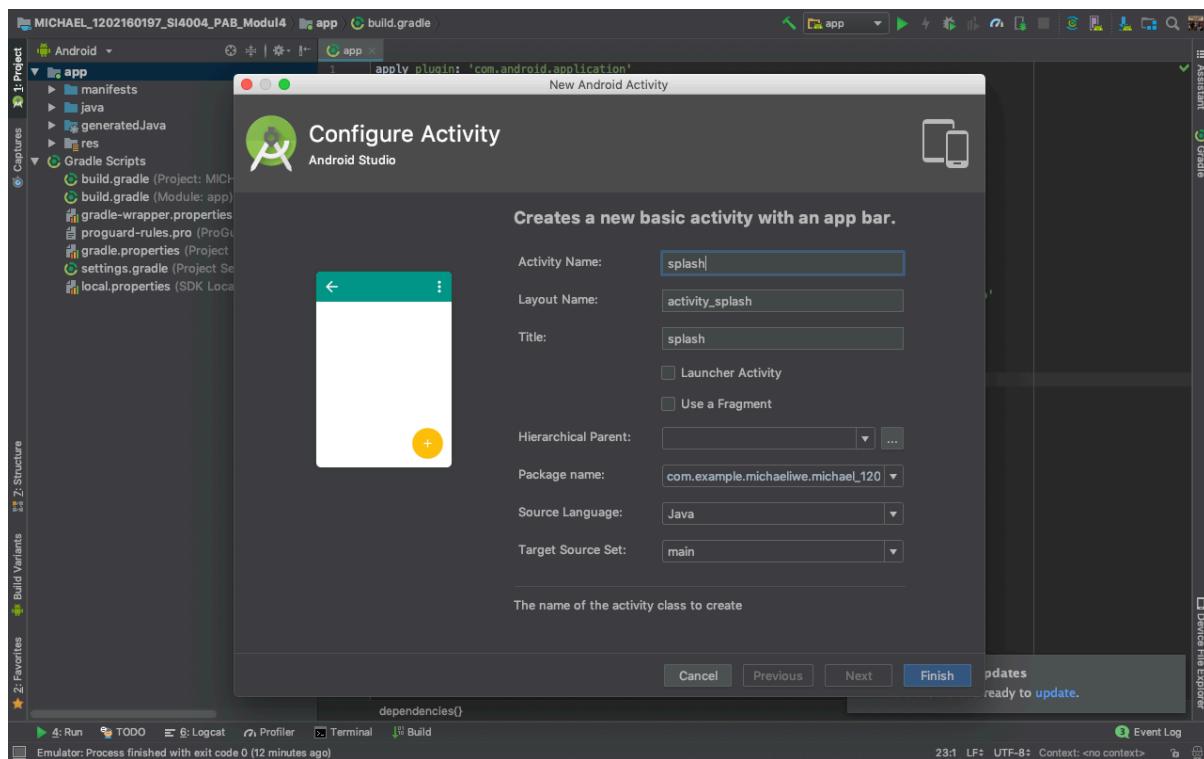
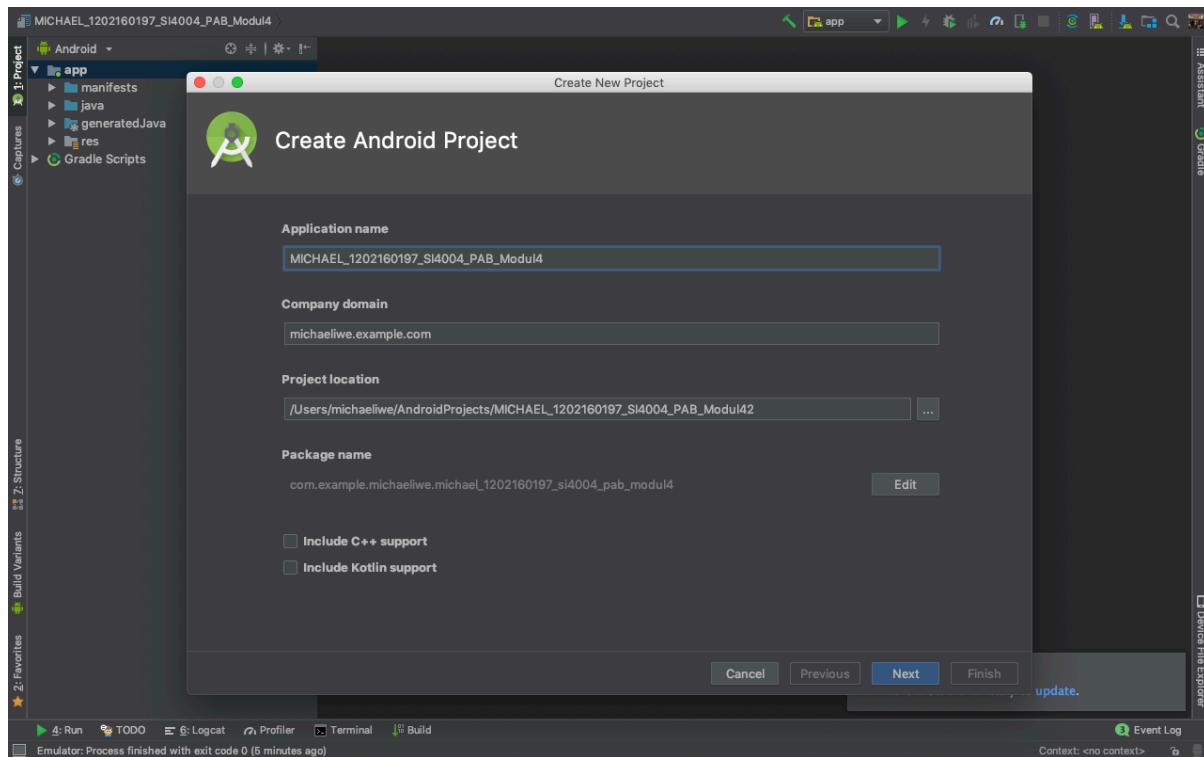
## DOKUMENTASI PRAKTIKUM MODUL 04

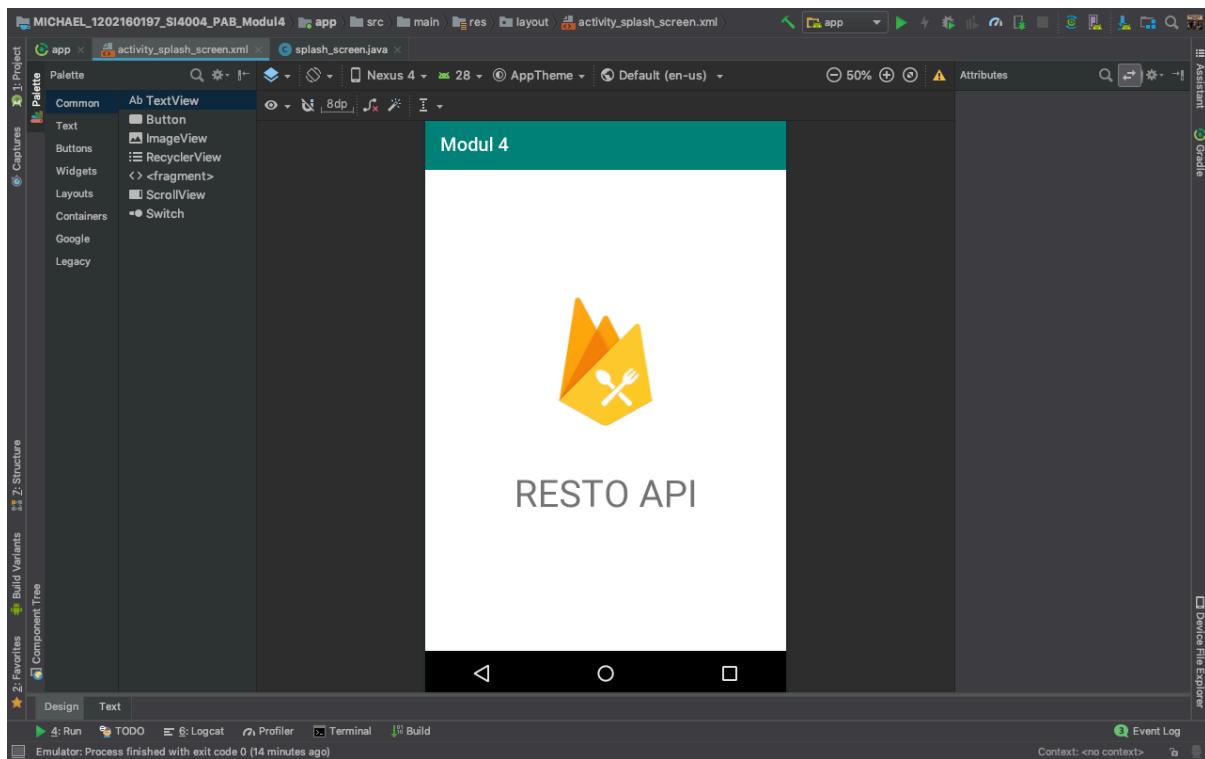
### PEMOGRAMAN APLIKASI BERGERAK

Nama : Michael Addryan Liwe

NIM : 1202160197

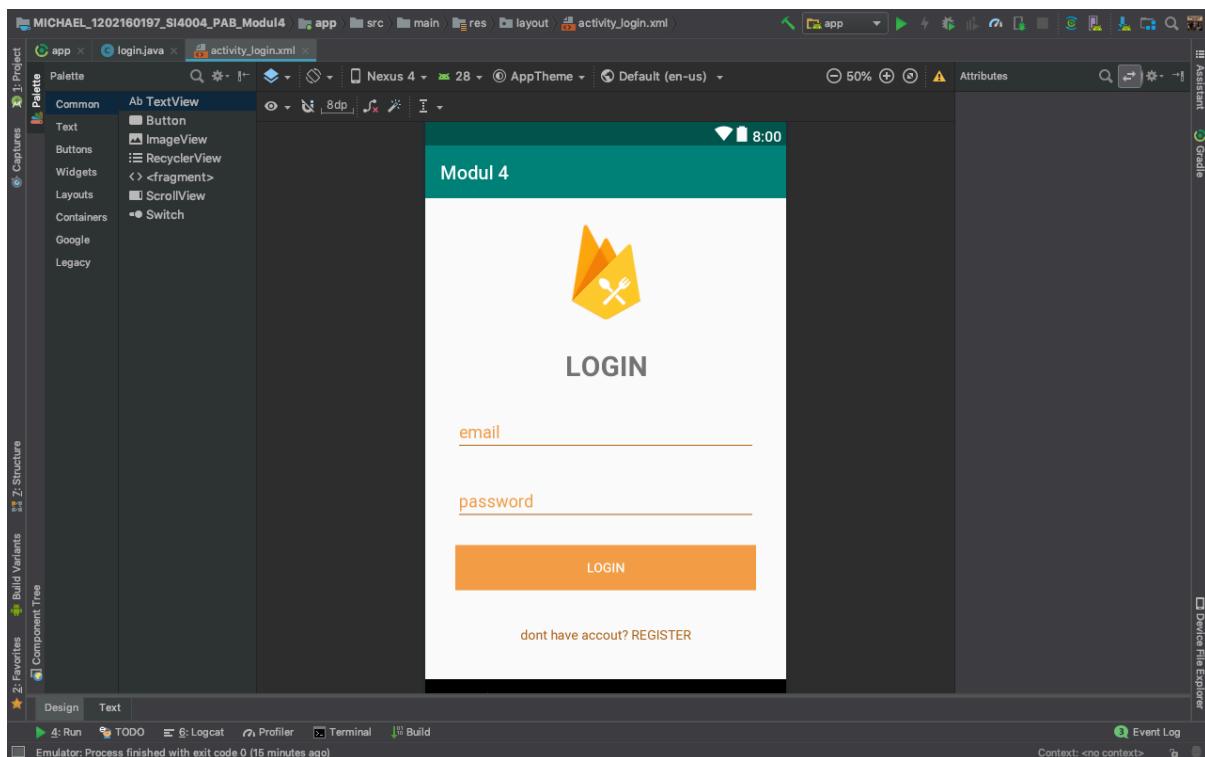
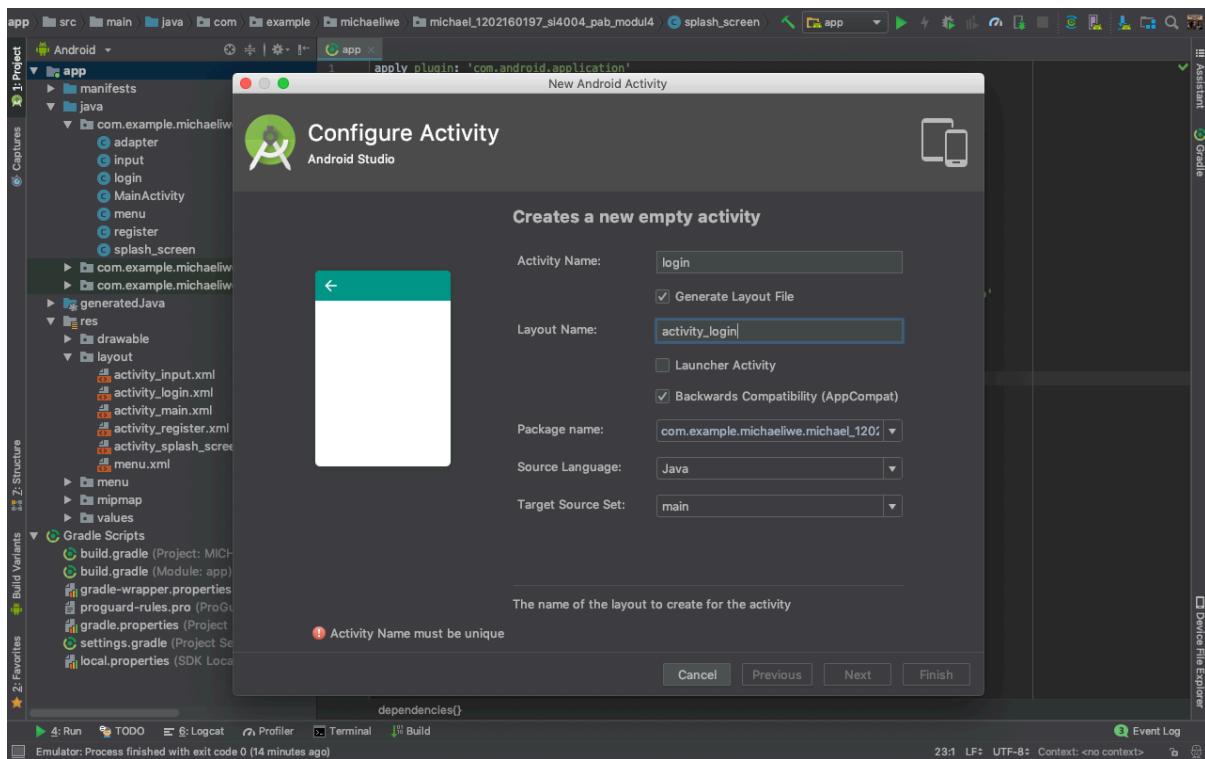
Kelas : SI4004





The screenshot shows the Android Studio interface with the following details:

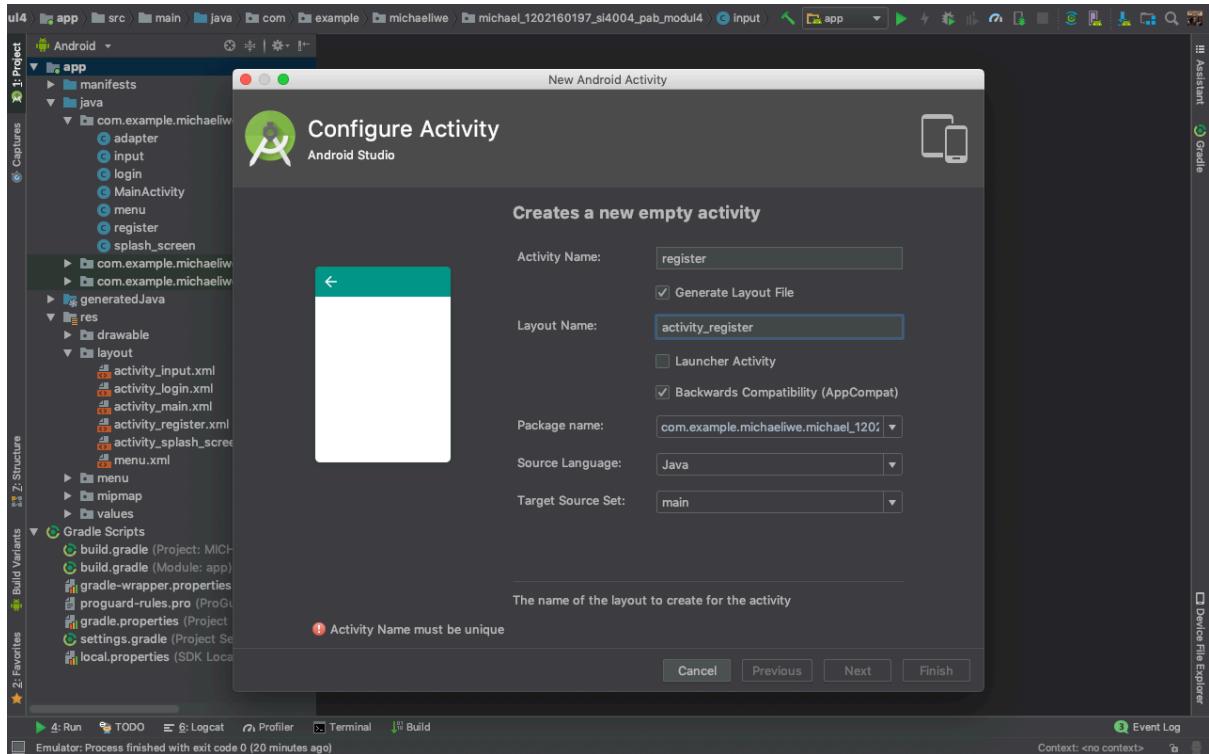
- Title Bar:** app, activity\_splash\_screen.xml, splash\_screen.java
- Toolbars:** App, activity\_splash\_screen.xml, splash\_screen.java
- Left Sidebar:** Project (app), Captures, Z-Structure, Build Variants, Favorites.
- Code Editor:** Java code for splash\_screen.java:package com.example.michaeliwe.michael\_1202160197\_si4004\_pab\_modul4;  
import ...  
public class splash\_screen extends AppCompatActivity {  
 public static final String TAG = "SplashScreen";  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 setContentView(R.layout.activity\_splash\_screen);  
  
 final Handler handler = new Handler();  
 handler.postDelayed(() -> {  
 startActivity(new Intent(getApplicationContext(), login.class));  
 finish();  
 }, delayMillis: 5000L);  
 }  
}
- Bottom Navigation:** Run, TODO, Logcat, Profiler, Terminal, Build.
- Status Bar:** Emulator: Process finished with exit code 0 (14 minutes ago)

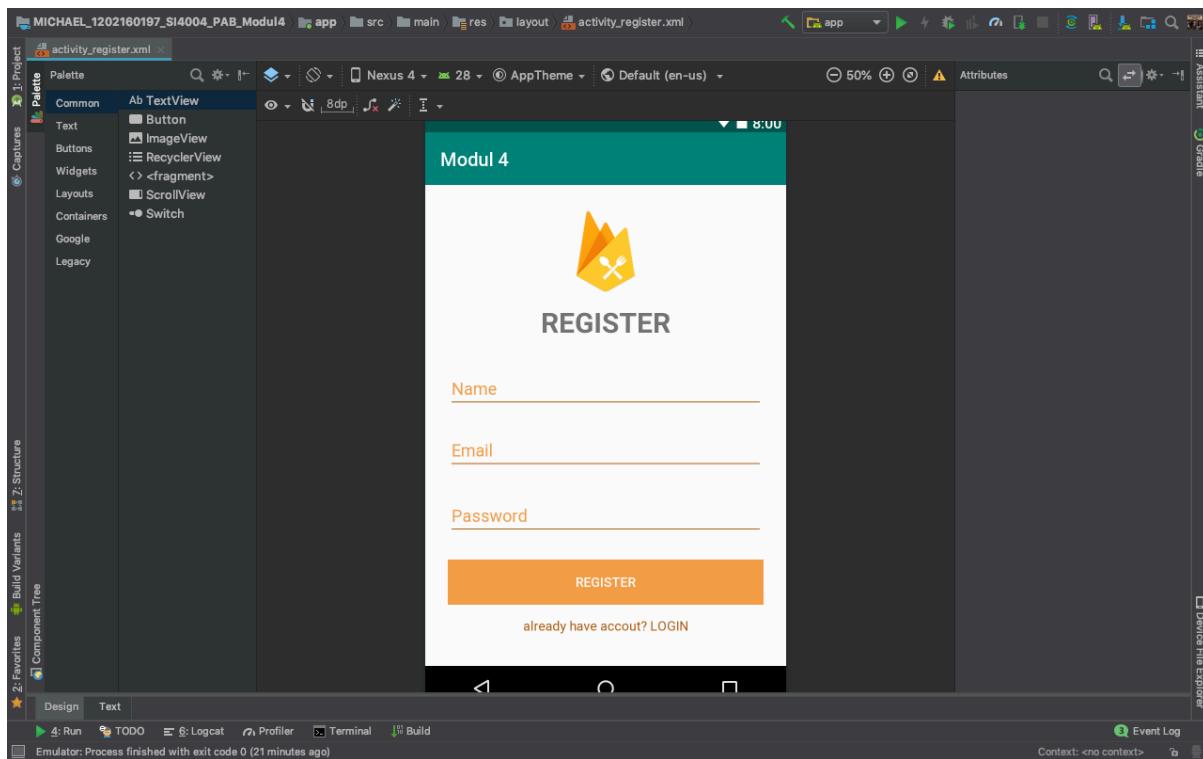


```
public void login(View view){
    if (check()) {
        new AsyncTask<Void,Void,Boolean>(){
            @Override
            protected Boolean doInBackground(Void... voids) {
                mAuth.signInWithEmailAndPassword(email.getText().toString(), password.getText().toString())
                    .addOnCompleteListener(task) ->
                if (task.isSuccessful()){
                    startActivity(new Intent(getApplicationContext(), MainActivity.class));
                    finish();
                } else {
                    Toast.makeText(context, task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                }
            }
            return null;
        }
        @Override
        protected void onPostExecute(Boolean aBoolean) { super.onPostExecute(aBoolean); }
    }.execute();
}

public void goToRegister(View view){
    startActivity(new Intent(getApplicationContext(), register.class));
    finish();
}

public boolean check(){
    if (email.getText().toString().equals("")){
        buat_toast("tolong isi dahulu kolom email");
        email.setText("");
        return false;
    }
    if (password.getText().toString().equals("")){
        buat_toast("tolong isi dahulu kolom password");
        password.setText("");
        return false;
    }
}
login > login() > new AsyncTask
```

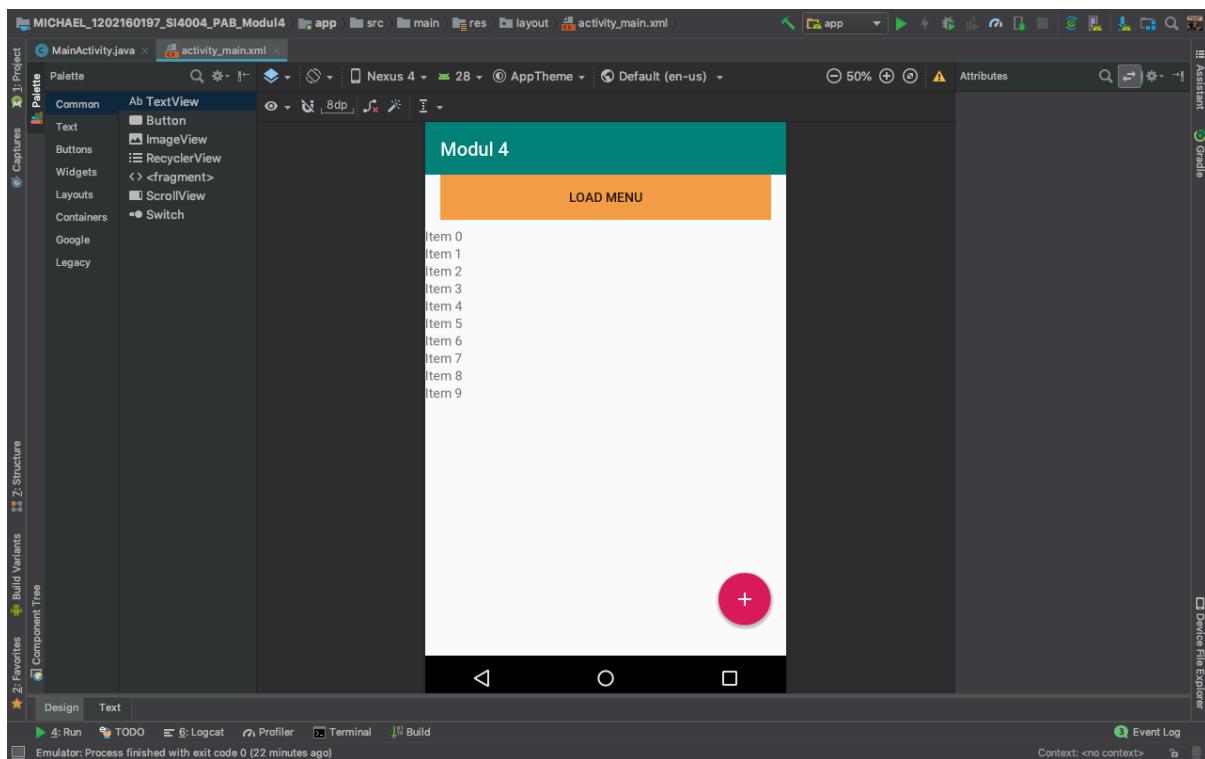
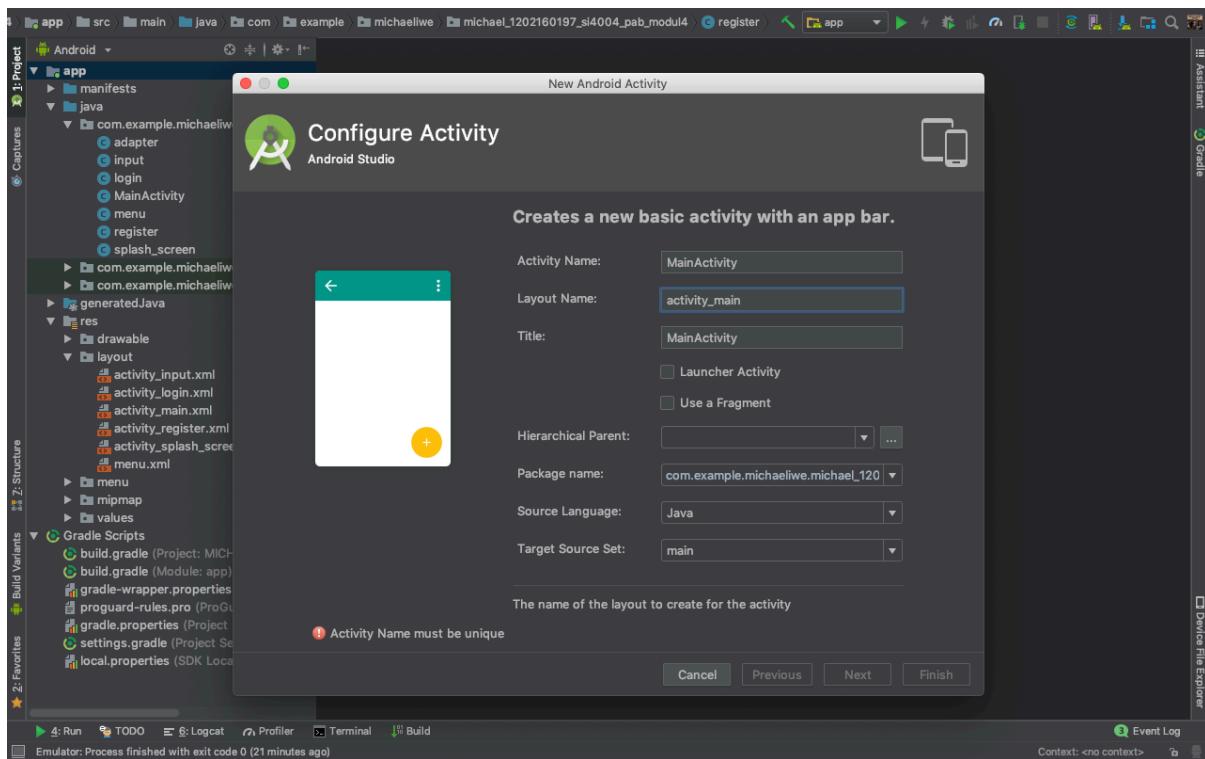


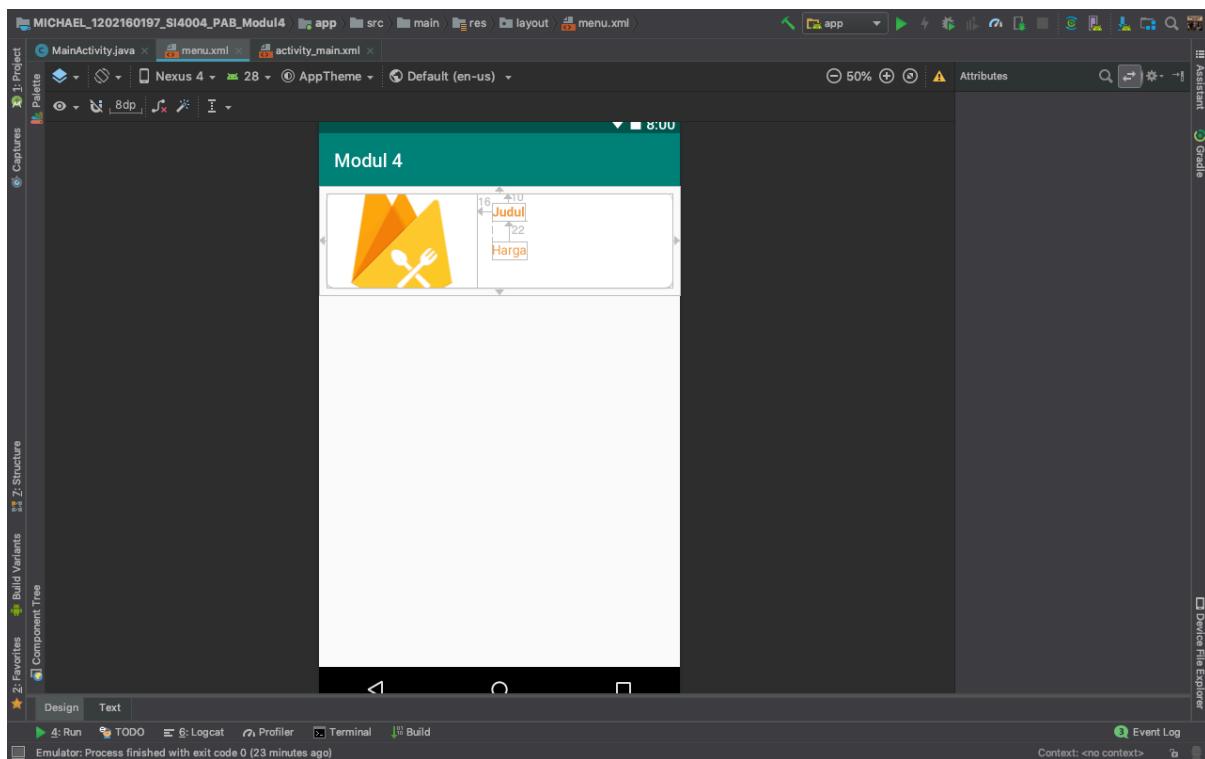
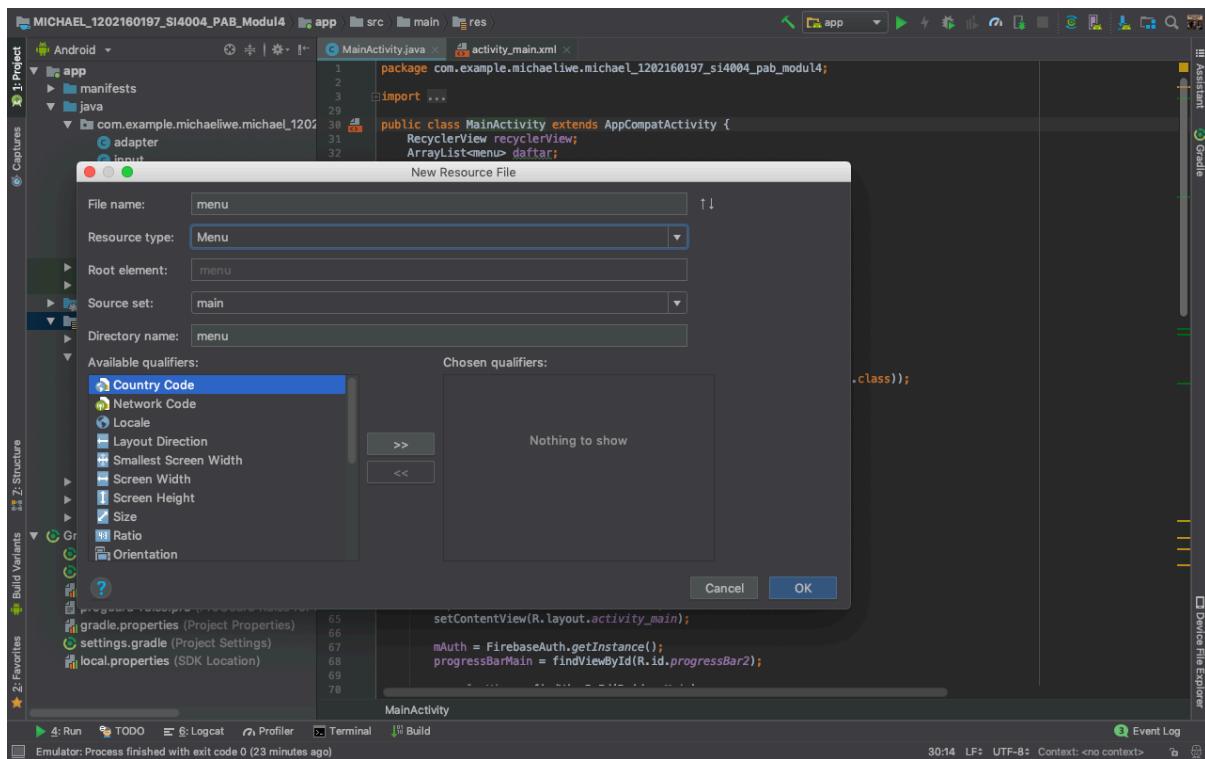


The screenshot shows the Android Studio interface with two files open:

- activity\_register.xml**: The XML layout file for the registration screen, containing views for name, email, password, and a register button.
- register.java**: The Java code for the registration logic. It includes methods for registering users and navigating to the login screen. It also contains a `check()` method to validate input fields.

```
32     }
33
34     public void register(View view){
35         if (check()){
36             mAuth = FirebaseAuth.getInstance();
37             mAuth.createUserWithEmailAndPassword(email.getText().toString(),password.getText().toString())
38                 .addOnCompleteListener(task) -> {
39                     if (task.isSuccessful()){
40                         FirebaseUser user = mAuth.getCurrentUser();
41                         UserProfileChangeRequest userProfileChangeRequest = new UserProfileChangeRequest.Builder().setDisplayName(name.getText().toString()).buat
42
43                         user.updateProfile(userProfileChangeRequest);
44                         startActivity(new Intent( packageContext: register.this,login.class));
45                         finish();
46                     } else {
47                         buat_toast( isi: "Register gagal");
48                     }
49                 });
50
51     }
52
53 }
54
55 public void goToLogin(View view){
56     startActivity(new Intent( packageContext: register.this,login.class));
57     finish();
58 }
59
60 public boolean check(){
61     if (name.getText().toString().equals("")){
62         buat_toast( isi: "Tolong isi dahulu kolom name");
63         password.setText("");
64         return false;
65     }
66     if (email.getText().toString().equals ""){
67         buat_toast( isi: "Tolong isi dahulu kolom email");
68         email.setText("");
69         return false;
70     }
71     if (password.getText().toString().equals ""){
72         buat_toast( isi: "Tolong isi dahulu kolom password");
73         password.setText("");
74         return false;
75     }
76     return true;
77 }
78 }
```





MainActivity.java

```
1 package com.example.michaeliwe.michael_1202160197_si4004_pab_modul4;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6     RecyclerView recyclerView;
7     ArrayList<menu> daftar;
8     adapter adapterCard;
9     FirebaseFirestore db;
10    FirebaseAuth mAuth;
11    ProgressBar progressBarMain;
12    int REQUEST_MENU = 404;
13
14    @Override
15    public boolean onCreateOptionsMenu(Menu menu) {
16        MenuInflater inflater = getMenuInflater();
17        inflater.inflate(R.menu.activity_main, menu);
18        return super.onCreateOptionsMenu(menu);
19    }
20
21    private void logout() {
22        mAuth.signOut();
23        startActivity(new Intent(getApplicationContext(), MainActivity.this, login.class));
24        finish();
25    }
26
27    @Override
28    public boolean onOptionsItemSelected(MenuItem item) {
29        switch (item.getItemId()) {
30            case R.id.logout:
31                logout();
32                break;
33        }
34        return super.onOptionsItemSelected(item);
35    }
36
37    @Override
38    protected void onCreate(Bundle savedInstanceState) {
39        super.onCreate(savedInstanceState);
40        setContentView(R.layout.activity_main);
41
42        mAuth = FirebaseAuth.getInstance();
43        progressBarMain = findViewById(R.id.progressBar2);
44
45        recyclerView = findViewById(R.id.rvMain);
46        daftar = new ArrayList<>();
47        recyclerView.setLayoutManager(new LinearLayoutManager(context: MainActivity.this));
48        adapterCard = new adapter(daftar, mContext: MainActivity.this);
49        recyclerView.setAdapter(adapterCard);
50        db = FirebaseFirestore.getInstance();
51
52        init();
53    }
54
55    public void add(View view) {
56        startActivityForResult(new Intent(getApplicationContext(), MainActivity.this, input.class), REQUEST_MENU);
57    }
58
59    @SuppressLint("StaticFieldLeak")
60    public void init() {
61        daftar.clear();
62        progressBarMain.setVisibility(View.VISIBLE);
63        new AsyncTask<Void, Void, Void>() {
64            @Override
65            protected void onPreExecute() { super.onPreExecute(); }
66
67            @Override
68            protected void onPostExecute(Void aVoid) { super.onPostExecute(aVoid); }
69
70            @Override
71            protected Void doInBackground(Void... voids) {
72
73                getthat();
74                return null;
75            }
76        }.execute();
77    }
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99 }
```

MainActivity

Emulator: Process finished with exit code 0 (24 minutes ago)

MainActivity.java

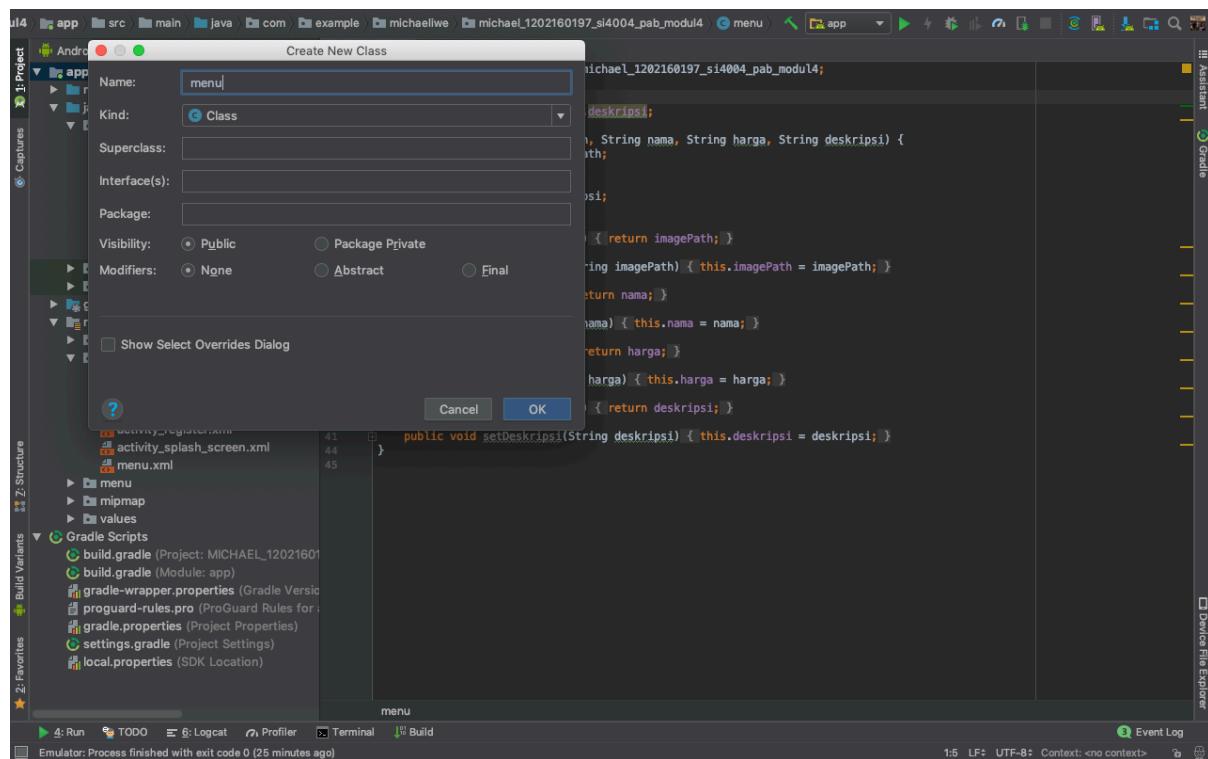
```
1 package com.example.michaeliwe.michael_1202160197_si4004_pab_modul4;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6     RecyclerView recyclerView;
7     ArrayList<menu> daftar;
8     adapter adapterCard;
9     FirebaseFirestore db;
10    FirebaseAuth mAuth;
11    ProgressBar progressBarMain;
12    int REQUEST_MENU = 404;
13
14    @Override
15    protected void onCreate(Bundle savedInstanceState) {
16        super.onCreate(savedInstanceState);
17        setContentView(R.layout.activity_main);
18
19        mAuth = FirebaseAuth.getInstance();
20        progressBarMain = findViewById(R.id.progressBar2);
21
22        recyclerView = findViewById(R.id.rvMain);
23        daftar = new ArrayList<>();
24        recyclerView.setLayoutManager(new LinearLayoutManager(context: MainActivity.this));
25        adapterCard = new adapter(daftar, mContext: MainActivity.this);
26        recyclerView.setAdapter(adapterCard);
27        db = FirebaseFirestore.getInstance();
28
29        init();
30    }
31
32    public void add(View view) {
33        startActivityForResult(new Intent(getApplicationContext(), MainActivity.this, input.class), REQUEST_MENU);
34    }
35
36    @SuppressLint("StaticFieldLeak")
37    public void init() {
38        daftar.clear();
39        progressBarMain.setVisibility(View.VISIBLE);
40        new AsyncTask<Void, Void, Void>() {
41            @Override
42            protected void onPreExecute() { super.onPreExecute(); }
43
44            @Override
45            protected void onPostExecute(Void aVoid) { super.onPostExecute(aVoid); }
46
47            @Override
48            protected Void doInBackground(Void... voids) {
49
50                getthat();
51                return null;
52            }
53        }.execute();
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 }
```

MainActivity

Emulator: Process finished with exit code 0 (24 minutes ago)

The screenshot shows the Android Studio interface with the code editor open to `MainActivity.java`. The code implements a background task to fetch data from a Firestore collection and update a RecyclerView. The `onPostExecute` method handles the result, and `doInBackground` contains the logic to query the database. The `loadMenu` and `getthat` methods are also defined.

```
95
96
97     @Override
98     protected void onPostExecute(Void aVoid) { super.onPostExecute(aVoid); }
99
100    @Override
101    protected Void doInBackground(Void... voids) {
102
103        getthat();
104        return null;
105    }
106    .execute();
107
108 }
109
110 public void loadMenu(View view) { init(); }
111
112 public void getthat() {
113     db.collection("auth.getUid()).get().addOnCompleteListener(task) -> {
114         if (task.isSuccessful()) {
115             for (QueryDocumentSnapshot document : task.getResult()) {
116
117                 Log.d("Result", "msg: " + document.getId() + " => " + document.getData());
118                 daftar.add(new menu(document.getId(), document.get("NamaMenu").toString(), document.get("Harga").toString(), document.get("Deskripsi").toString()));
119
120             }
121             adapterCard.notifyDataSetChanged();
122
123         }
124     }
125     progressBarMain.setVisibility(View.GONE);
126
127 });
128
129 }
130
131
132
133
134     @Override
135     protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
136         init();
137         super.onActivityResult(requestCode, resultCode, data);
138     }
139
140 }
```



The screenshot shows the Android Studio interface with the Java code for 'menu.java' in the center. The code defines a class 'menu' with fields for imagePath, nama, harga, and deskripsi, and methods for setting and getting these values. The project structure on the left includes 'app', 'res', 'values', and 'Gradle Scripts' sections.

```

1 package com.example.michaeliwe.michael_1202160197_si4004_pab_modul4;
2
3 public class menu {
4     String imagePath,nama,harga,deskripsi;
5
6     public menu(String imagePath, String nama, String harga, String deskripsi) {
7         this.imagePath = imagePath;
8         this.nama = nama;
9         this.harga = harga;
10        this.deskripsi = deskripsi;
11    }
12
13    public String getImagePath() { return imagePath; }
14
15    public void setImagePath(String imagePath) { this.imagePath = imagePath; }
16
17    public String getNama() { return nama; }
18
19    public void setNama(String nama) { this.nama = nama; }
20
21    public String getHarga() { return harga; }
22
23    public void setHarga(String harga) { this.harga = harga; }
24
25    public String getDeskripsi() { return deskripsi; }
26
27    public void setDeskripsi(String deskripsi) { this.deskripsi = deskripsi; }
28
29}
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

```

The screenshot shows the 'Create New Class' dialog box in the foreground, with 'Name:' set to 'adapater' and 'Kind:' set to 'Class'. The background shows the Java code for 'adapter.java' which extends RecyclerView.Adapter<adapter.ViewHolder>. The code handles onCreateViewHolder, onBindViewHolder, and getItemCount methods, along with a ViewHolder class that implements View.OnClickListener.

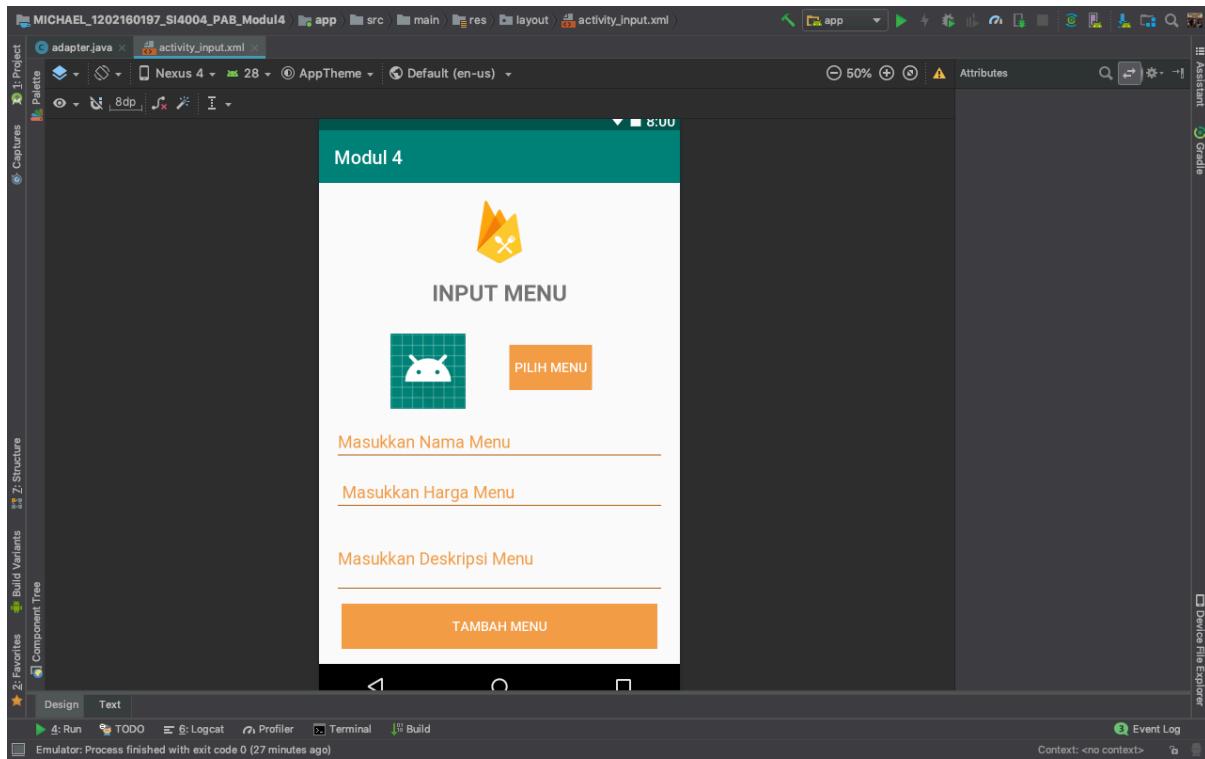
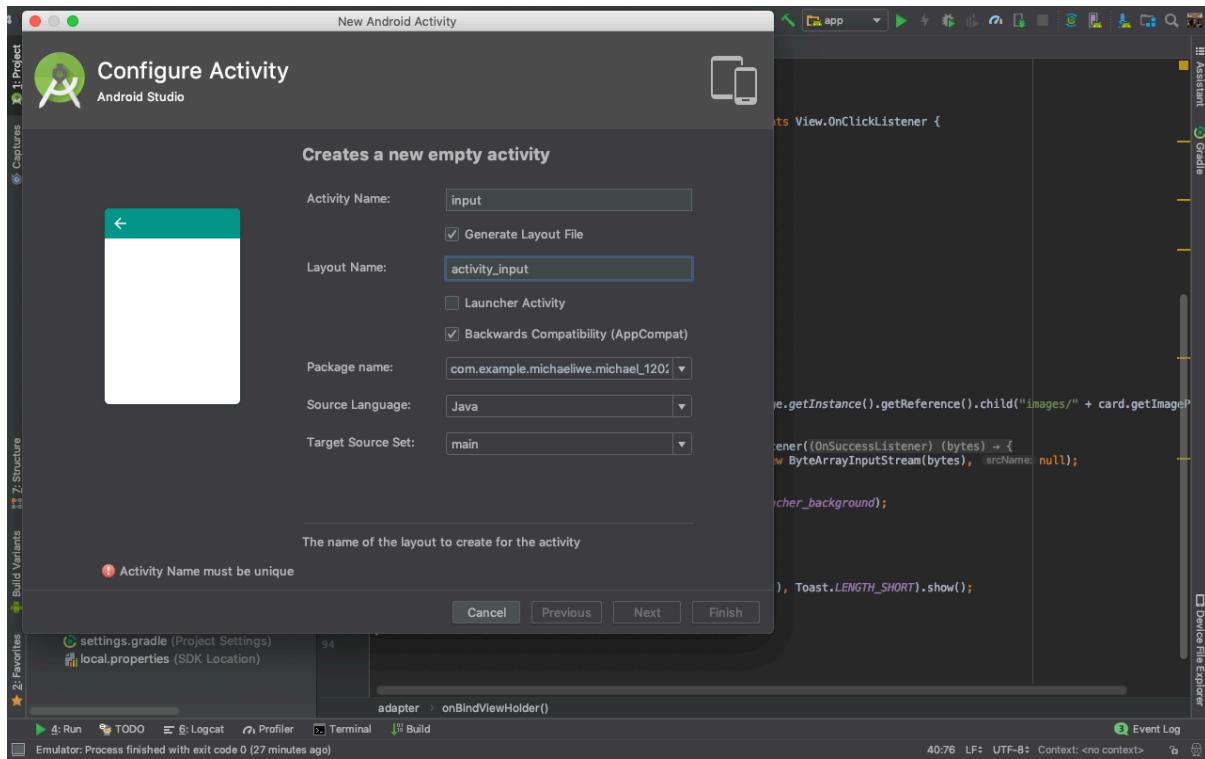
```

1 package iwe.michael_1202160197_si4004_pab_modul4;
2
3 import android.view.View;
4 import android.widget.TextView;
5 import android.widget.ImageView;
6
7 import androidx.recyclerview.widget.RecyclerView;
8 import androidx.recyclerview.widget.RecyclerView.ViewHolder;
9
10 import com.example.michaeliwe.michael_1202160197_si4004_pab_modul4.R;
11
12 public class adapter extends RecyclerView.Adapter<adapter.ViewHolder> {
13
14     RecyclerView.ViewHolder daftarChat;
15
16     t<menu> daftarChat, Context mContext) {
17         afterChat;
18         intext;
19
20         ateViewHolder(ViewGroup parent, int viewType) {
21             ViewHolder LayoutInflator.from(mContext).inflate(R.layout.menu, parent,
22                 attachToRoot: false));
23
24
25         older(adapter.ViewHolder holder, int position) {
26             hat.get(position);
27
28
29         @Override
30         public int getItemCount() { return daftarChat.size(); }
31
32         class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
33             private TextView nama, harga;
34             private ImageView image;
35
36             public ViewHolder(@NotNull View itemView) {
37                 super(itemView);
38
39                     nama = itemView.findViewById(R.id.txJudulCard);
40                     harga = itemView.findViewById(R.id.txHarga);
41                     image = itemView.findViewById(R.id.cardImg);
42
43                     itemView.setOnClickListener(this);
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

```

```
1 package com.example.michaeliwe.michael_1202160197_si4004_pab_modul4;
2
3 import ...
4
5 public class adapter extends RecyclerView.Adapter<adapter.ViewHolder>{
6     private ArrayList<menu> daftarChat;
7     private Context mContext;
8
9     public adapter(ArrayList<menu> daftarChat, Context mContext) {
10         this.daftarChat = daftarChat;
11         this.mContext = mContext;
12     }
13
14     @Override
15     public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
16         return new adapter.ViewHolder(LayoutInflater.from(mContext).inflate(R.layout.menu, parent, attachToRoot: false));
17     }
18
19     @Override
20     public void onBindViewHolder(adapter.ViewHolder holder, int position) {
21         menu card = daftarChat.get(position);
22         holder.bindTo(card);
23     }
24
25     @Override
26     public int getItemCount() { return daftarChat.size(); }
27
28     class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
29         private TextView nama, harga;
30         private ImageView image;
31
32         public ViewHolder(@NotNull View itemView) {
33             super(itemView);
34
35             nama = itemView.findViewById(R.id.txJudulCard);
36             harga = itemView.findViewById(R.id.txHarga);
37             image = itemView.findViewById(R.id.cardImg);
38
39             itemView.setOnClickListener(this);
40         }
41
42         @Override
43         public void onClick(View v) {
44             Toast.makeText(mContext, nama.getText().toString(), Toast.LENGTH_SHORT).show();
45         }
46     }
47 }
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
```

```
1 package com.example.michaeliwe.michael_1202160197_si4004_pab_modul4;
2
3 import ...
4
5 public class adapter extends RecyclerView.Adapter<adapter.ViewHolder>{
6     private ArrayList<menu> daftarChat;
7     private Context mContext;
8
9     public adapter(ArrayList<menu> daftarChat, Context mContext) {
10         this.daftarChat = daftarChat;
11         this.mContext = mContext;
12     }
13
14     @Override
15     public int getItemCount() { return daftarChat.size(); }
16
17     class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
18         private TextView nama, harga;
19         private ImageView image;
20
21         public ViewHolder(@NotNull View itemView) {
22             super(itemView);
23
24             nama = itemView.findViewById(R.id.txJudulCard);
25             harga = itemView.findViewById(R.id.txHarga);
26             image = itemView.findViewById(R.id.cardImg);
27
28             itemView.setOnClickListener(this);
29         }
30
31         @Override
32         public void onClick(View v) {
33             Toast.makeText(mContext, nama.getText().toString(), Toast.LENGTH_SHORT).show();
34         }
35
36         @Override
37         public void bindTo(final menu card) {
38             nama.setText(card.getNama());
39             harga.setText(card.getHarga());
40
41             final StorageReference islandRef = FirebaseStorage.getInstance().getReference().child("images/" + card.getImagePath());
42
43             final long ONE_MEGABYTE = 1024 * 1024;
44             islandRef.getBytes(ONE_MEGABYTE).addOnSuccessListener((OnSuccessListener<byte[]>) (bytes) -> {
45                 Drawable d = Drawable.createFromStream(new ByteArrayInputStream(bytes), null);
46                 image.setImageDrawable(d);
47             }).addOnFailureListener(exception -> {
48                 image.setImageResource(R.drawable.ic_launcher_background);
49             });
50         }
51     }
52
53     @Override
54     public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
55         return new adapter.ViewHolder(LayoutInflater.from(mContext).inflate(R.layout.menu, parent, attachToRoot: false));
56     }
57
58     @Override
59     public void onBindViewHolder(adapter.ViewHolder holder, int position) {
60         menu card = daftarChat.get(position);
61         holder.bindTo(card);
62     }
63
64     @Override
65     public int getItemCount() { return daftarChat.size(); }
66
67     class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
68         private TextView nama, harga;
69         private ImageView image;
70
71         public ViewHolder(@NotNull View itemView) {
72             super(itemView);
73
74             nama = itemView.findViewById(R.id.txJudulCard);
75             harga = itemView.findViewById(R.id.txHarga);
76             image = itemView.findViewById(R.id.cardImg);
77
78             itemView.setOnClickListener(this);
79         }
80
81         @Override
82         public void onClick(View v) {
83             Toast.makeText(mContext, nama.getText().toString(), Toast.LENGTH_SHORT).show();
84         }
85     }
86
87 }
88
89
90
91
92
93
94
```



```
1 package com.example.michaeliwe.michael_1202160197_si4004_pab_modul4;
2
3 import ...
4
5 public class input extends AppCompatActivity {
6     int REQUEST = 91, REQUEST_GET_SINGLE_FILE = 202, REQUEST_CAPTURE_IMAGE = 234;
7     Bitmap bitmap;
8     ImageView foto;
9     FirebaseFirestore db;
10    FirebaseAuth mAuth;
11    Uri url;
12    String imagePath;
13    EditText nama, harga, desk;
14
15    FirebaseStorage storage;
16    StorageReference storageReference;
17
18    Map<String, Object> menu;
19
20    ProgressBar progressBar;
21
22    boolean success = false;
23
24    @Override
25    protected void onCreate(Bundle savedInstanceState) {
26        super.onCreate(savedInstanceState);
27        setContentView(R.layout.activity_input);
28        foto = findViewById(R.id.img);
29        db = FirebaseFirestore.getInstance();
30        mAuth = FirebaseAuth.getInstance();
31
32        storage = FirebaseStorage.getInstance();
33        storageReference = storage.getReference();
34        nama = findViewById(R.id.edNamaInput);
35        harga = findViewById(R.id.edHargaInput);
36        desk = findViewById(R.id.edDeskInput);
37
38        progressBar = findViewById(R.id.progressBar);
39
40        FirebaseFirestoreSettings settings = new FirebaseFirestoreSettings.Builder()
41            .setPersistenceEnabled(true)
42            .build();
43
44        db.setFirestoreSettings(settings);
45
46        tambahMenu();
47    }
48
49    private void tambahMenu() {
50        new AsyncTask<Void, Boolean, Boolean>() {
51
52            @Override
53            protected void onPostExecute() {
54                progressBar.setVisibility(View.VISIBLE);
55                menu = new HashMap<>();
56
57                super.onPostExecute();
58            }
59
60            @Override
61            protected void onPostExecute(Boolean aBoolean) {
62                if (aBoolean) {
63                    Toast.makeText(context, input.this, text: "Berhasil", Toast.LENGTH_SHORT).show();
64                    finish();
65                }
66                super.onPostExecute(aBoolean);
67            }
68
69            @Override
70            protected Boolean doInBackground(Void... voids) {
71                menu.put("NamaMenu", nama.getText().toString());
72                menu.put("Harga", harga.getText().toString());
73                menu.put("Deskripsi", desk.getText().toString());
74                menu.put("Date", new SimpleDateFormat().format(new Date()));
75
76                db.collection(mAuth.getUid())
77                    .add(menu)
78                    .addOnSuccessListener((OnSuccessListener) (documentReference) -> {
79                        docRef = documentReference.getId();
80                        //Upload data berhasil;
81                        success = true;
82                        if (url != null) {
83                            StorageReference ref = storageReference.child("images/" + documentReference.getId());
84                            ref.putFile(url)
85                                .addOnSuccessListener((OnSuccessListener) (taskSnapshot) -> {
86                                    Toast.makeText(context, input.this, text: "UpLoaded", Toast.LENGTH_SHORT).show();
87                                });
88                        }
89                    });
90                }
91            }
92        }.execute();
93    }
94
95    @Override
96    public void onBackPressed() {
97        super.onBackPressed();
98    }
99}
```

```
100    }
101
102    @Override
103    protected void onPostExecute(Boolean aBoolean) {
104        if (aBoolean) {
105            Toast.makeText(context, input.this, text: "Berhasil", Toast.LENGTH_SHORT).show();
106            finish();
107        }
108        super.onPostExecute(aBoolean);
109    }
110
111    @Override
112    protected Boolean doInBackground(Void... voids) {
113        menu.put("NamaMenu", nama.getText().toString());
114        menu.put("Harga", harga.getText().toString());
115        menu.put("Deskripsi", desk.getText().toString());
116        menu.put("Date", new SimpleDateFormat().format(new Date()));
117
118        db.collection(mAuth.getUid())
119            .add(menu)
120            .addOnSuccessListener((OnSuccessListener) (documentReference) -> {
121                docRef = documentReference.getId();
122                //Upload data berhasil;
123                success = true;
124                if (url != null) {
125                    StorageReference ref = storageReference.child("images/" + documentReference.getId());
126                    ref.putFile(url)
127                        .addOnSuccessListener((OnSuccessListener) (taskSnapshot) -> {
128                            Toast.makeText(context, input.this, text: "UpLoaded", Toast.LENGTH_SHORT).show();
129                        });
130                }
131            });
132    }
133}
```

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays Java code for a class named 'input'. The code implements an interface for adding items to a menu and handles file uploads to Firestore. It uses Firebase's Storage API to upload files and Firestore's database API to store document references. The code includes logic for handling file selection, starting an AsyncTask, and updating a progress bar.

```
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
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  
788  
789  
789  
790  
791  
792  
793  
794  
795  
796  
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  
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  
1088  
1089  
1089  
1090  
1091  
1092  
1093  
1094  
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  
1149  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1159  
1160  
1161  
1162
```

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays Java code for a class named 'input'. The code implements an interface for adding items to a menu and handles file selection from the gallery. It uses Intent's ACTION\_GET\_CONTENT to start a file selection activity and handle the result. The code includes logic for checking permissions and starting the activity for result.

```
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
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  
788  
789  
789  
790  
791  
792  
793  
794  
795  
796  
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  
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  
1088  
1089  
1089  
1090  
1091  
1092  
1093  
1094  
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  
1149  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
11
```

```
protected void onActivityResult(int requestCode, int resultCode, final Intent data) {
    if (resultCode == RESULT_OK) {
        if (requestCode == REQUEST_CAPTURE_IMAGE) {
            if (data != null && data.getExtras() != null) {
                new AsyncTask<Void, Void, Void>() {
                    @Override
                    protected void doInBackground(Void... voids) {
                        bitmap = (Bitmap) data.getExtras().get("data");
                        return null;
                    }

                    @Override
                    protected void onPostExecute(Void aVoid) {
                        foto.setImageBitmap(bitmap);
                        super.onPostExecute(aVoid);
                    }
                }.execute();
            }
        } else if (requestCode == REQUEST_GET_SINGLE_FILE) {
            new AsyncTask<Void, Void, Void>() {
                @Override
                protected void onPreExecute() {
                    uri = data.getData();
                    super.onPreExecute();
                }

                @Override
                protected void onPostExecute(Void aVoid) {
                    foto.setImageBitmap(bitmap);
                    super.onPostExecute(aVoid);
                }

                @Override
                protected Void doInBackground(Void... voids) {
                    imagePath = getPathFromURI(getApplicationContext(), uri);
                    try {
                        bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), uri);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    return null;
                }
            }.execute();
        }
    }
}

private String getPathFromURI(Context context, Uri uri) {
    final boolean isKitKat = Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT;
    Log.i("tag", "URI", msg: uri + "");
    String result = uri + "";
    // DocumentProvider
    if (isKitKat && result.contains("media.documents")) {
        String[] ary = result.split( regex: "/");
        int length = ary.length;
        String imgary = ary[length - 1];
        final String[] dat = imgary.split( regex: "%3A");
        final String docId = dat[1];
        final String type = dat[0];
        Uri contentUri = null;
        if ("image".equals(type)) {
            contentUri = MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
        } else if ("video".equals(type)) {
        } else if ("audio".equals(type)) {
        }
        final String selection = "_id=?";
        final String[] selectionArgs = new String[]{dat[1]};
    }
    return getDataColumn(context, contentUri, selection, selectionArgs);
} else if ("content".equalsIgnoreCase(uri.getScheme())) {
    return getDataColumn(context, uri, selection, selectionArgs);
} else if ("file".equalsIgnoreCase(uri.getScheme())) {
    return uri.getPath();
}
return null;
}

public static String getDataColumn(Context context, Uri uri, String selection, String[] selectionArgs) {
    Cursor cursor = null;
    final String column = "_data";
    final String[] projection = {
        column
    };
    try {
        cursor = context.getContentResolver().query(uri, projection, selection, selectionArgs, sortOrder);
        if (cursor != null && cursor.moveToFirst()) {
            final int column_index = cursor.getColumnIndexOrThrow(column);
        }
    } finally {
        if (cursor != null)
            cursor.close();
    }
}
```

```
Cursor cursor = null;
final String column = "_data";
final String[] projection = {
    column
};
try {
    cursor = context.getContentResolver().query(uri, projection, selection, selectionArgs, sortOrder);
    if (cursor != null && cursor.moveToFirst()) {
        final int column_index = cursor.getColumnIndexOrThrow(column);
    }
} finally {
    if (cursor != null)
        cursor.close();
}
```

The screenshot shows the Android Studio interface with the project structure and code editor visible.

**Project Structure:**

- app
- manifests
- java
  - com.example.michaeliwe.michael\_1202
    - adapter
    - input
    - login
    - MainActivity
    - menu
    - register
    - splash\_screen
  - com.example.michaeliwe.michael\_1202
  - generatedJava
  - res
    - layout
      - activity\_input.xml
      - activity\_login.xml
      - activity\_main.xml
      - activity\_register.xml
      - activity\_splash\_screen.xml
    - menu.xml
  - values
- Gradle Scripts
  - build.gradle (Project: MICHAEL\_12021601)
  - build.gradle (Module: app)
  - gradle-wrapper.properties (Grade Version: 2.10)
  - proguard-rules.pro (ProGuard Rules for: app)
  - gradle.properties (Project Properties)
  - settings.gradle (Project Settings)
  - local.properties (SDK Location)

**Code Editor (Input.java):**

```
if ("Image".equals(type)) {
    contentUri = MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
} else if ("Video".equals(type)) {
} else if ("audio".equals(type)) {
}
final String selection = "_id=?";
final String[] selectionArgs = new String[]{datat[1]};
return getDataColumn(context, contentUri, selection, selectionArgs);
} else if ("content".equalsIgnoreCase(uri.getScheme())) {
    return getDataColumn(context, uri, selection, selectionArgs);
} else if ("file".equalsIgnoreCase(uri.getScheme())) {
    return uri.getPath();
}
return null;
}

public static String getDataColumn(Context context, Uri uri, String selection, String[] selectionArgs) {
    Cursor cursor = null;
    final String column = "_data";
    final String[] projection = {
        column
    };
    try {
        cursor = context.getContentResolver().query(uri, projection, selection, selectionArgs, sortOrder: null);
        if (cursor != null && cursor.moveToFirst()) {
            final int column_index = cursor.getColumnIndexOrThrow(column);
            return cursor.getString(column_index);
        }
    } finally {
        if (cursor != null)
            cursor.close();
    }
    return null;
}
```

**Bottom Bar:**

- Run
- TODO
- Logcat
- Profiler
- Terminal
- Build

**Event Log:**

Emulator: Process finished with exit code 0 (31 minutes ago)