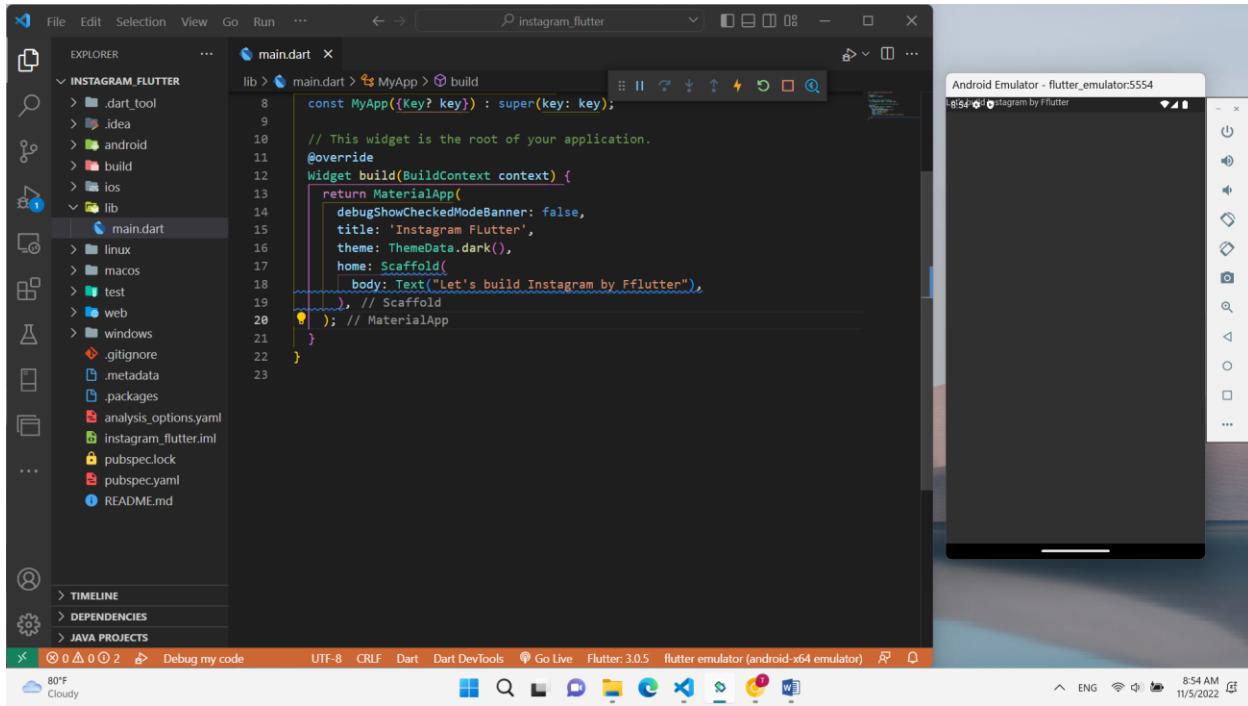
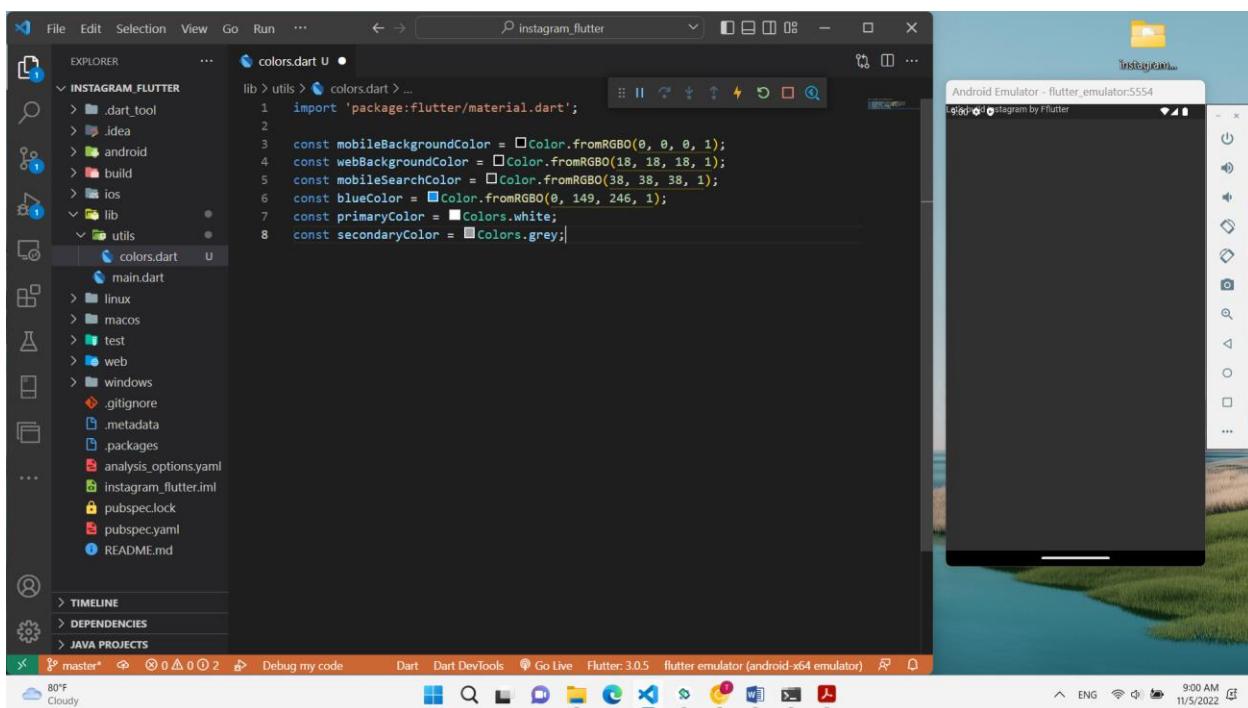


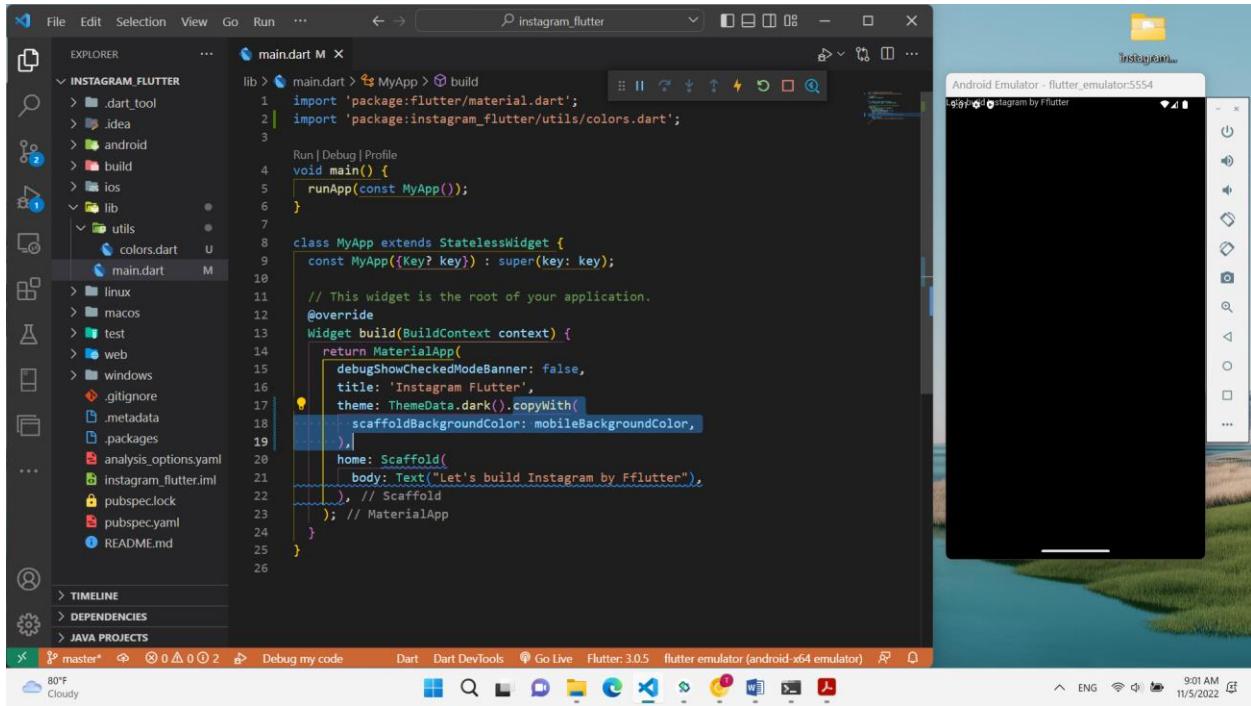
Bước 1. Khởi tạo dự án và thiết lập chủ đề



Bước 2. Tạo tập tin lib/utils và copy colors.dart vào để chỉnh màu sắc

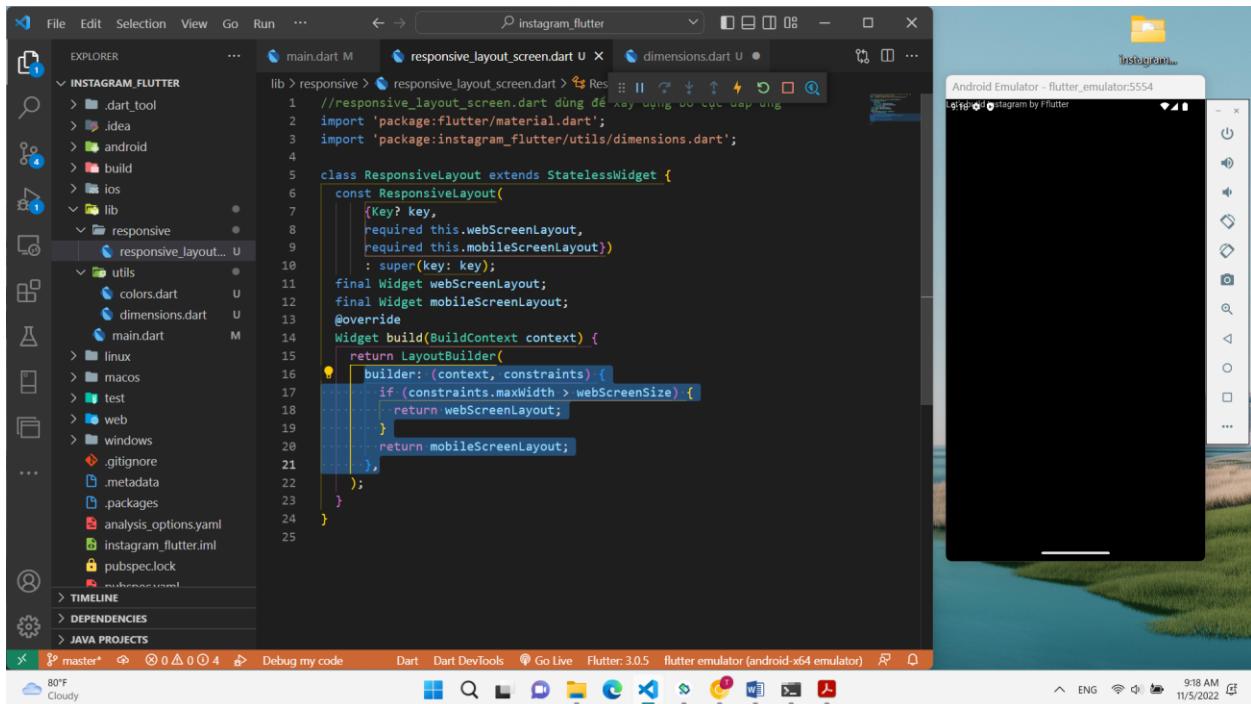


Bước 3. Hiệu chỉnh màu sắc

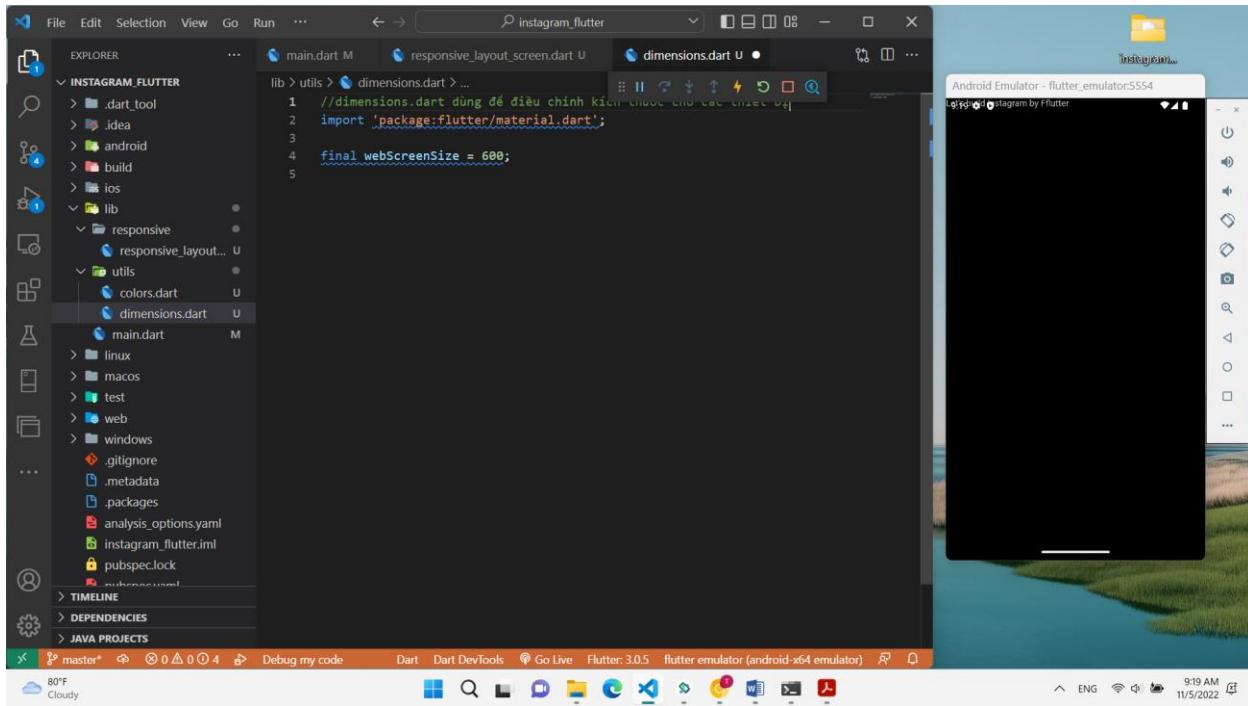


Bước 4. Xây dựng bố cục đáp ứng (Responsive layout screen)

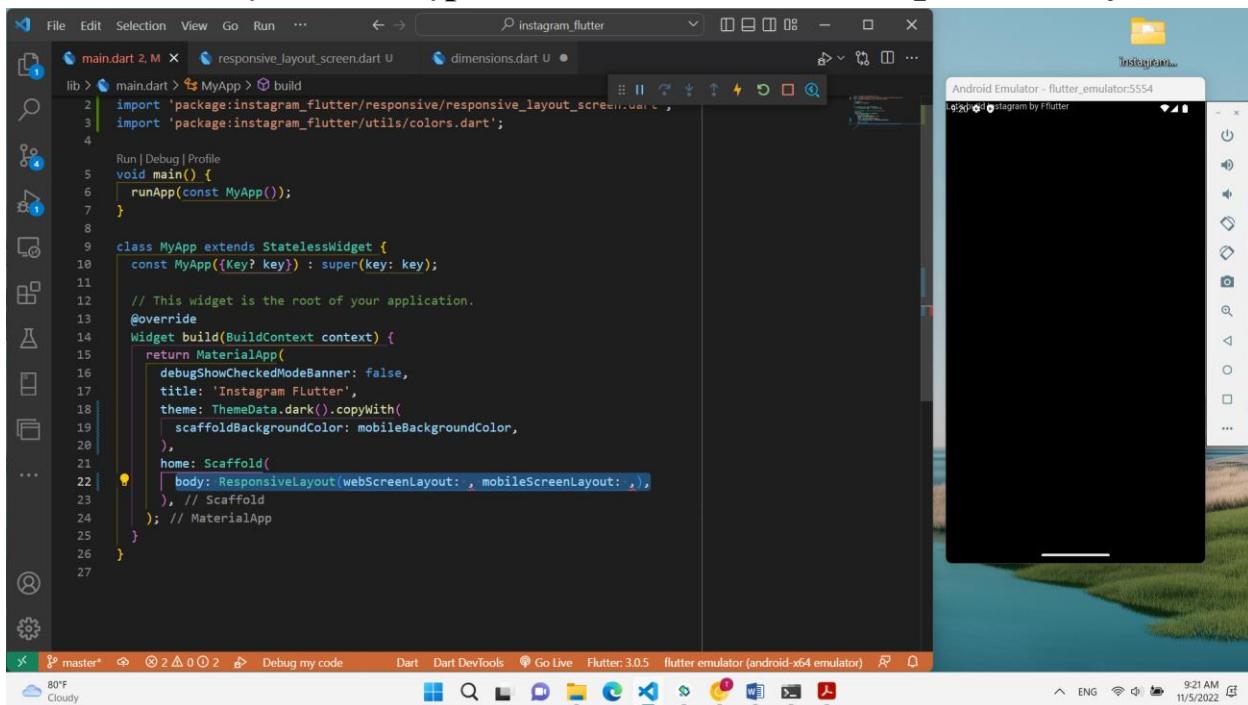
Tạo file lib/responsive/responsive_layout_screen.dart và chỉnh nội dung như hình bên dưới



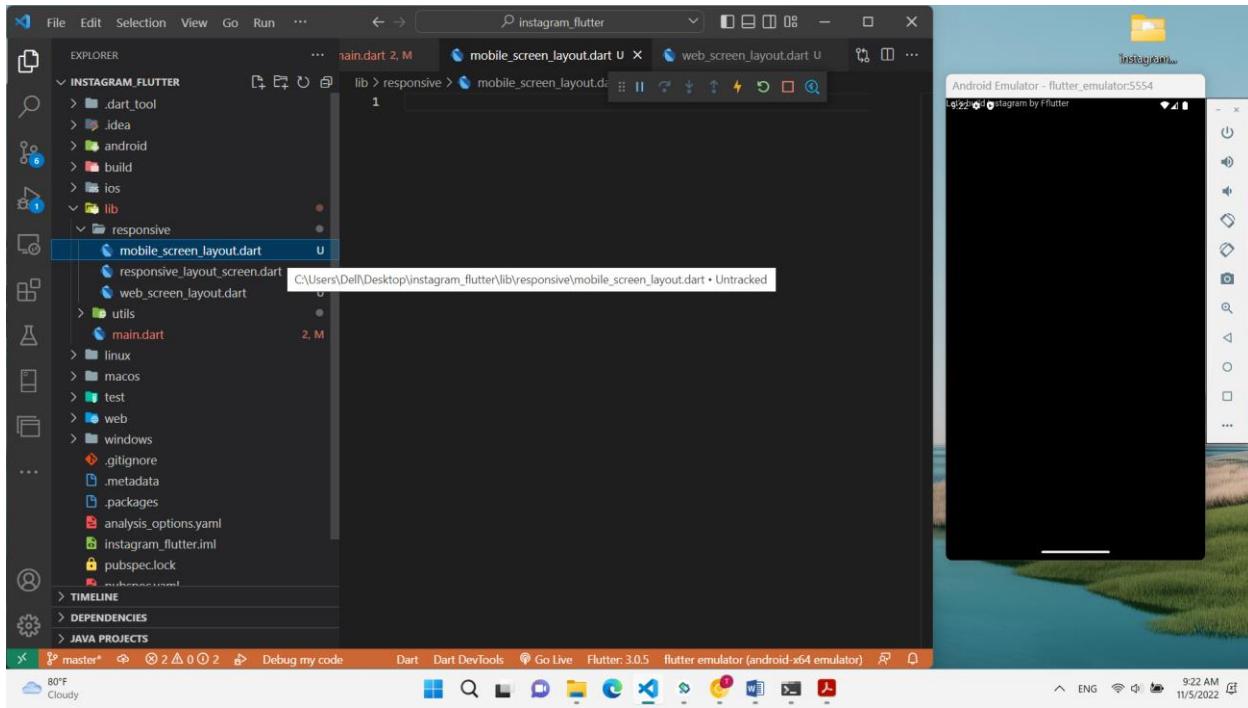
Bước 5. Tập tin dimensions.dart dùng để điều chỉnh kích thước cho các thiết bị



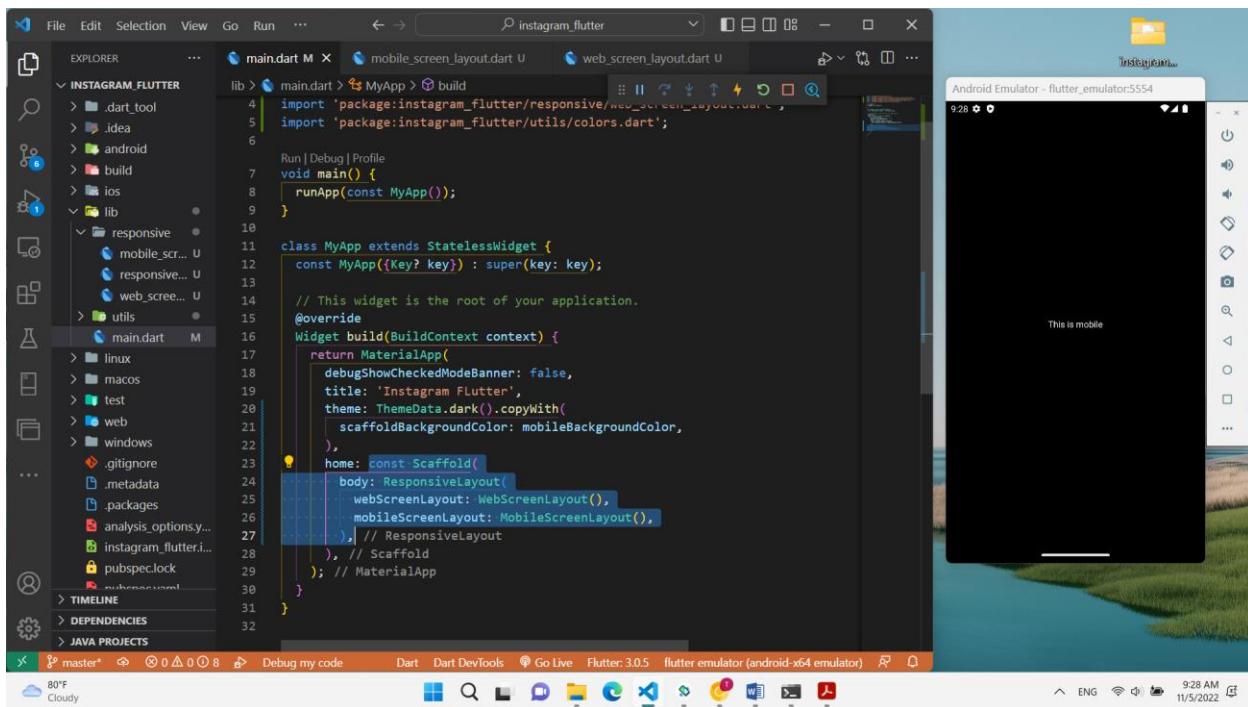
Bước 6. Hiệu chỉnh tập tin main.dart để thử responsive_layout



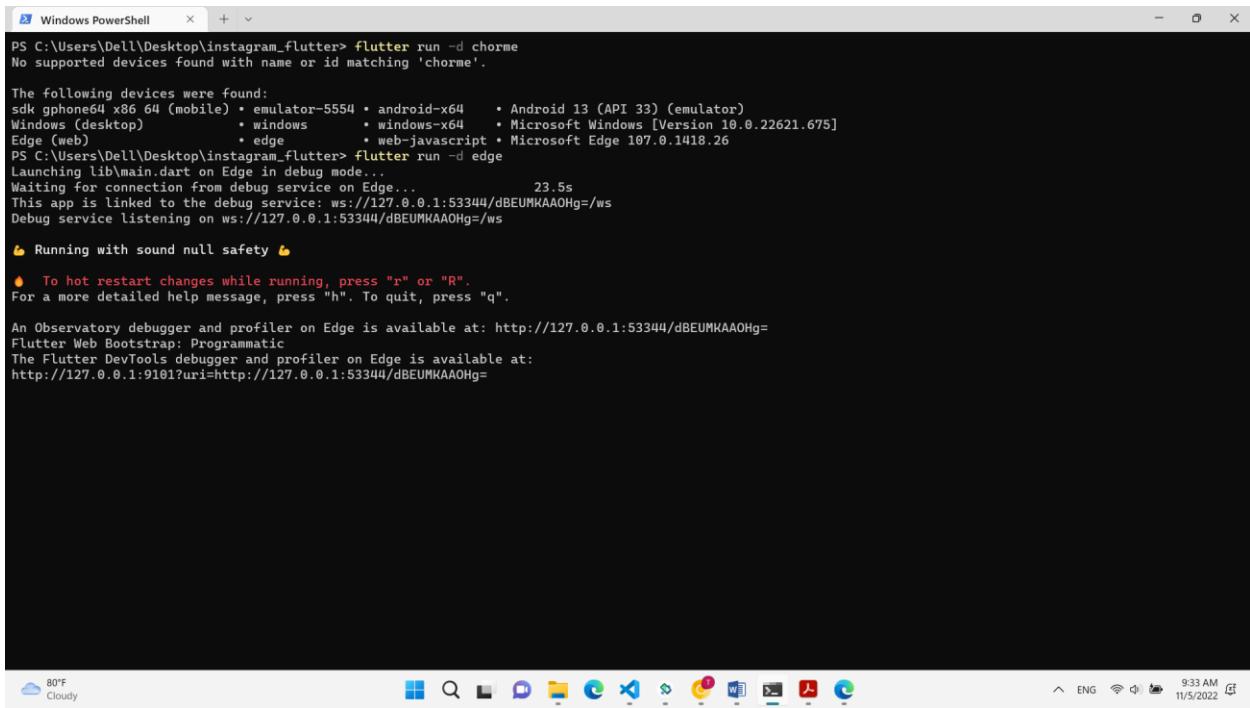
Bước 7. Trong lib/responsive tạo 2 tập tin mobile_screen_layout và web_screen_layout để tạo bộ cục đáp ứng cho di động và web



Bước 8. Test thử main.dart



Bước 9. Dùng lệnh flutter run -d edge để chạy trên web

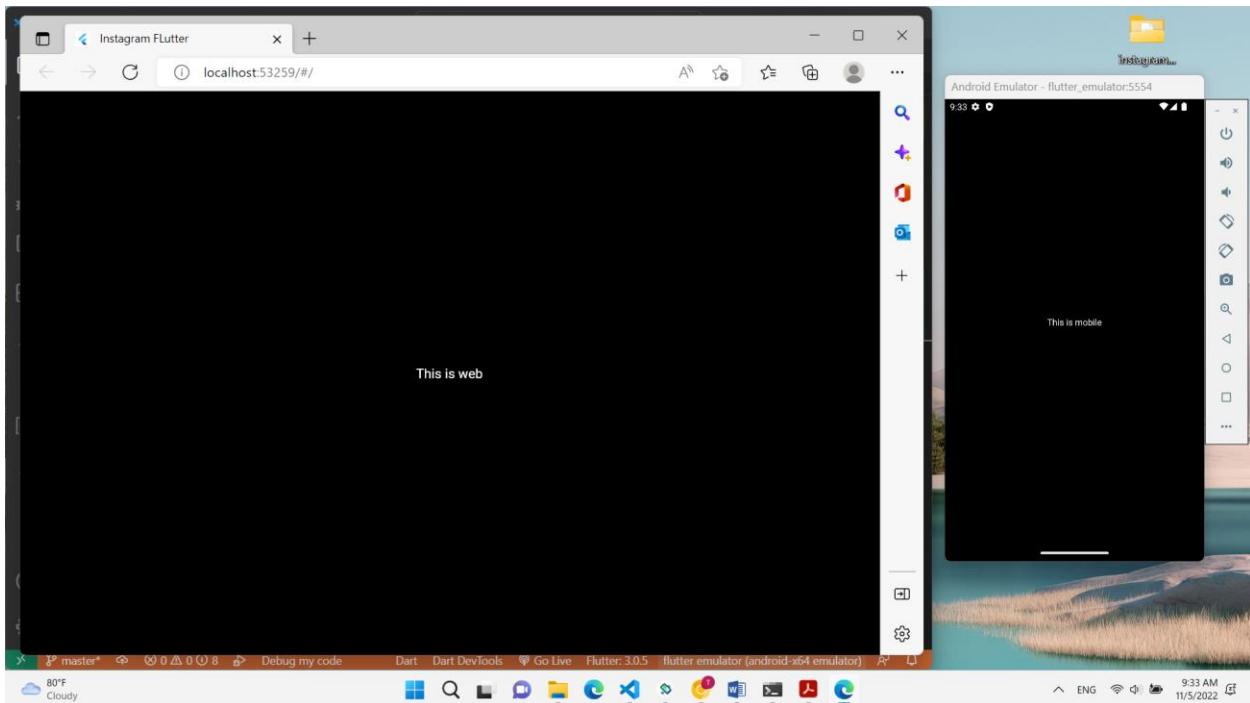


```
PS C:\Users\Huy\Documents\GitHub\Instagram_Flutter> flutter run -d edge
No supported devices found with name or id matching 'chromium'.
The following devices were found:
sdl gphone64 x86_64 (mobile) • emulator-5554 • android-x64      • Android 13 (API 33) (emulator)
Windows (desktop)           • windows       • windows-x64     • Microsoft Windows [Version 10.0.22621.675]
Edge (web)                  • edge          • web-javascript • Microsoft Edge 107.0.1418.26
PS C:\Users\Huy\Documents\GitHub\Instagram_Flutter> flutter run -d edge
Launching lib/main.dart on Edge in debug mode...
Waiting for connection from debug service on Edge...                          23.5s
This app is linked to the debug service: ws://127.0.0.1:53344/dBEUMKAAOHg=/ws
Debug service listening on ws://127.0.0.1:53344/dBEUMKAAOHg=/ws

Flutter Web Bootstrap: Programmatic
The Flutter DevTools debugger and profiler on Edge is available at:
http://127.0.0.1:9101?uri=http://127.0.0.1:53344/dBEUMKAAOHg=/ws

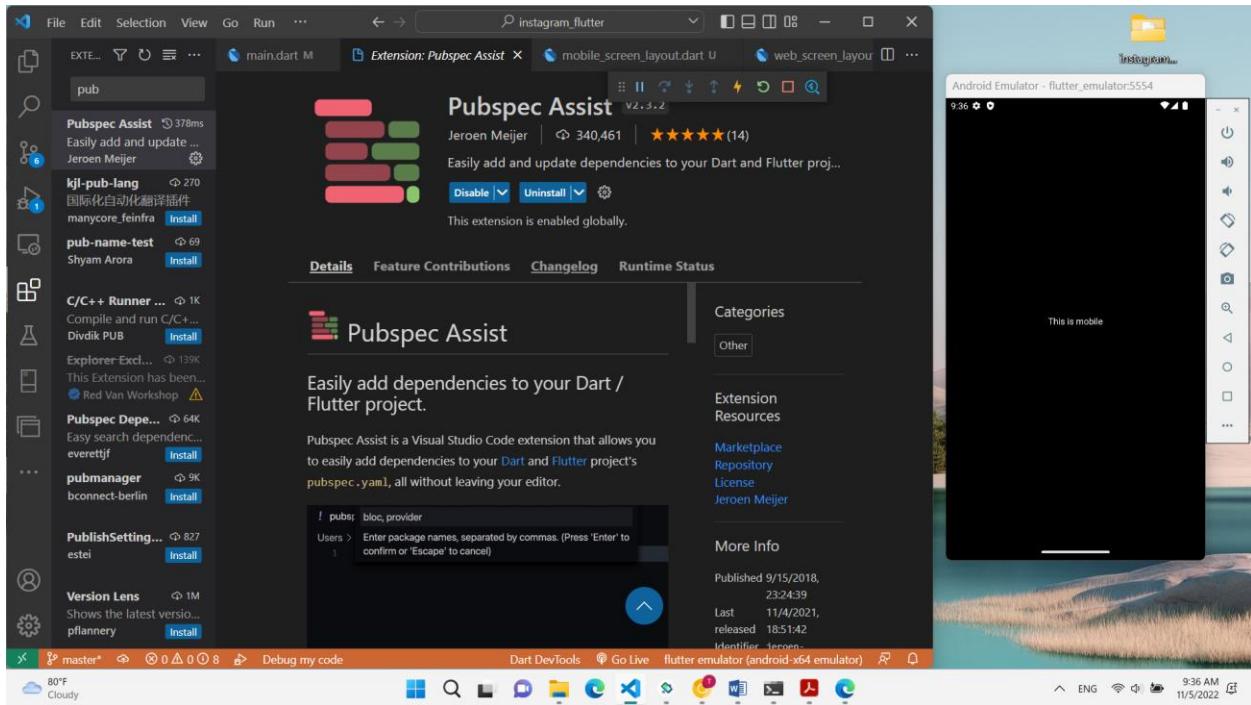
An Observatory debugger and profiler on Edge is available at: http://127.0.0.1:53344/dBEUMKAAOHg=
Flutter Web Bootstrap: Programmatic
The Flutter DevTools debugger and profiler on Edge is available at:
http://127.0.0.1:9101?uri=http://127.0.0.1:53344/dBEUMKAAOHg=/ws
```

Kết quả

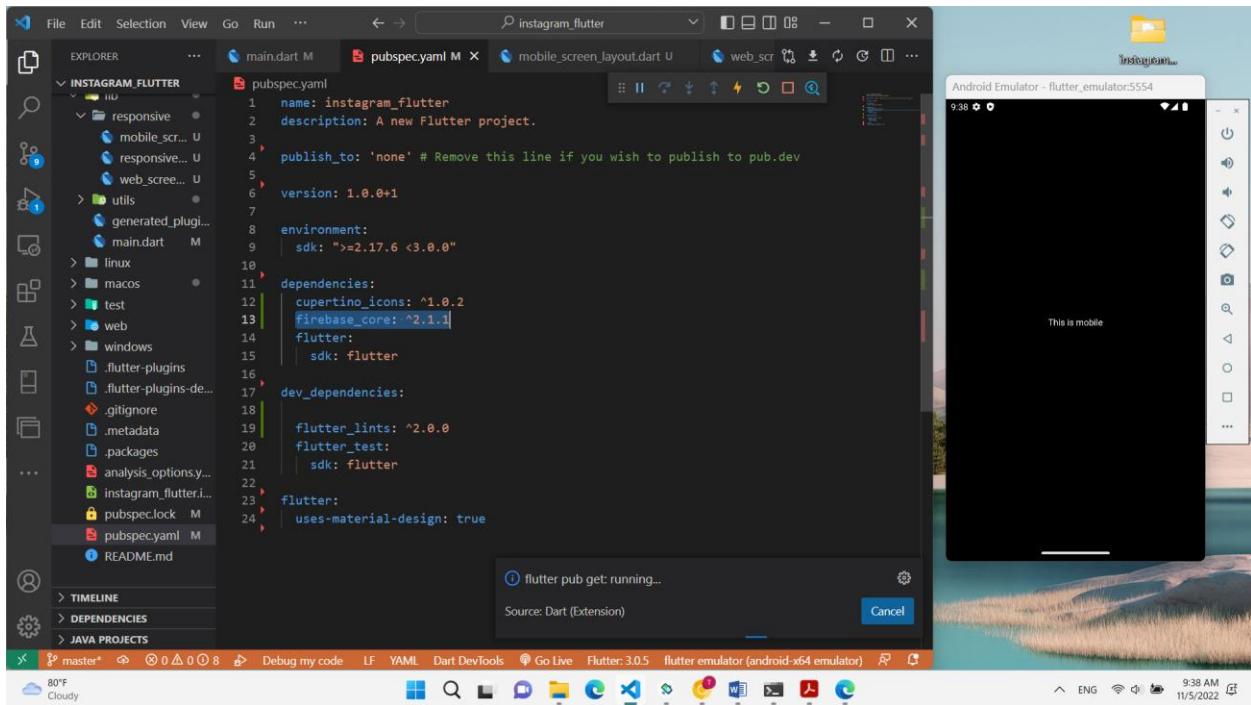


Bước 10. Cài đặt firebase

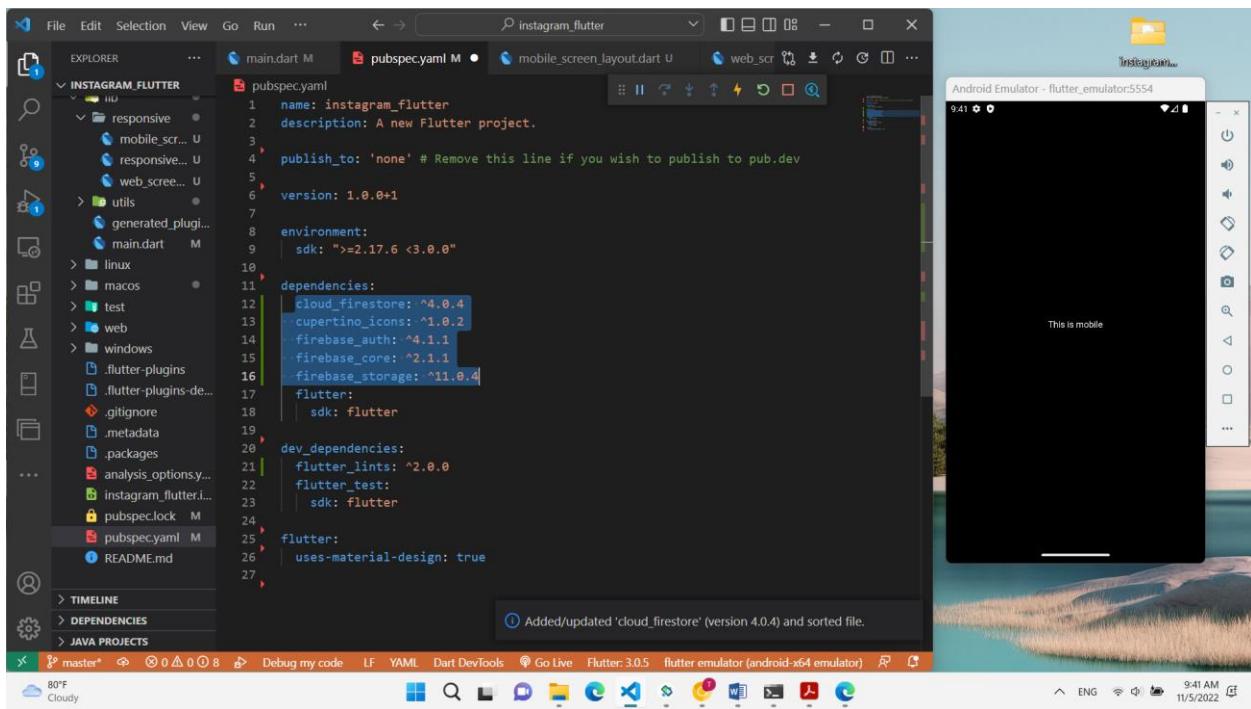
Cài đặt pubspec assist



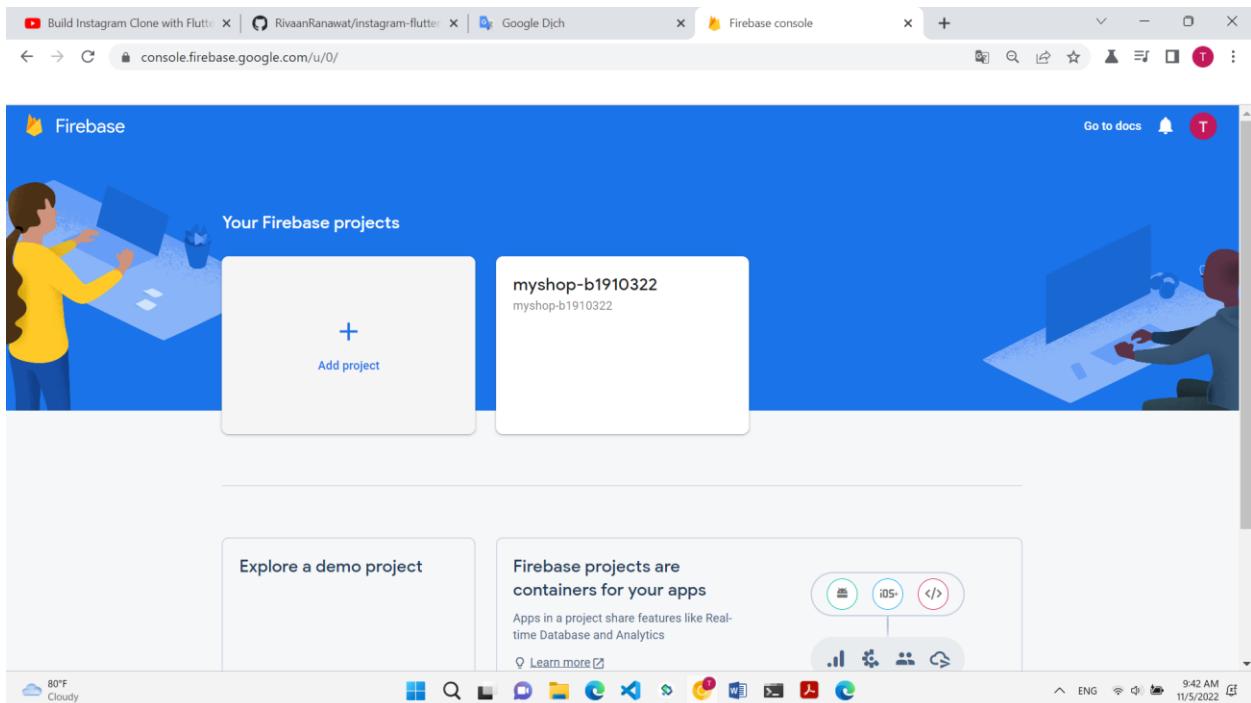
Cài đặt thành công firebase_score



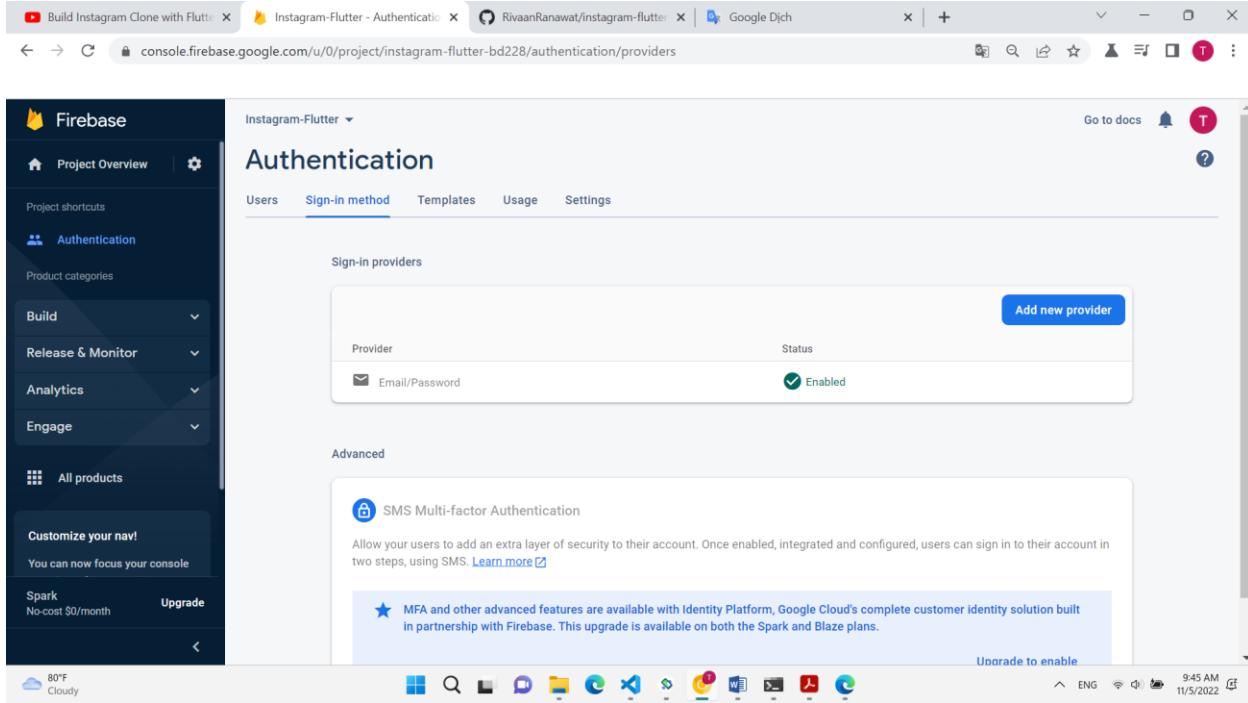
Cài đặt các dịch vụ firebase



Bước 11. Tạo project mới

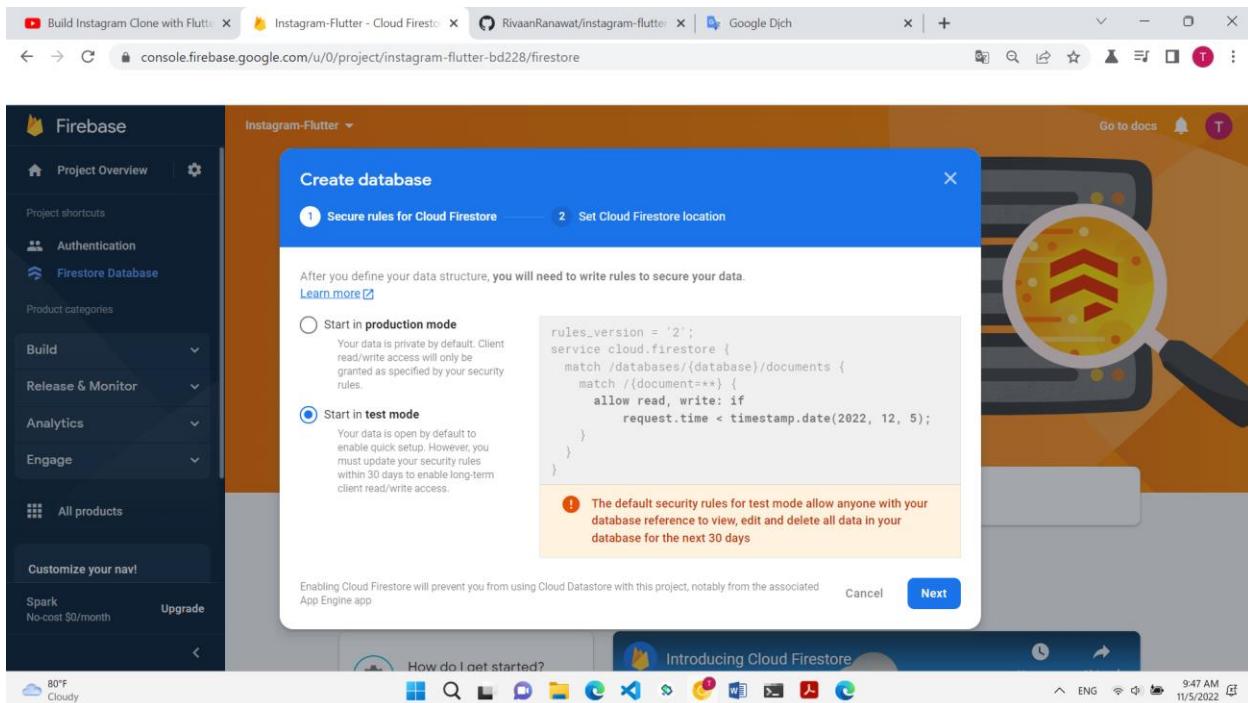


Bước 12. Sử dụng email/password



The screenshot shows the Firebase console's Authentication section. On the left sidebar, 'Authentication' is selected under 'Project Overview'. The main 'Sign-in method' tab is active. Under 'Sign-in providers', there is a table with one row: 'Email/Password' with a status of 'Enabled' and a green checkmark. Below this, the 'Advanced' section includes a 'SMS Multi-factor Authentication' card with a brief description and a note about MFA. A 'Upgrade to enable' button is visible at the bottom right of the card.

Bước 13. Tạo database



The screenshot shows the Firebase console's Cloud Firestore 'Create database' dialog. The dialog has two steps: 'Secure rules for Cloud Firestore' (selected) and 'Set Cloud Firestore location'. Step 1 displays security rules code and a note about test mode. Step 2 is partially visible. At the bottom, there is a note about preventing Cloud Datastore usage and a 'Next' button. The background shows a 'How do I get started?' banner and an 'Introducing Cloud Firestore' section.

The screenshot shows the Firebase Cloud Firestore Rules editor. On the left, there's a sidebar with project navigation and a weather widget. The main area has tabs for Data, Rules, Indexes, and Usage. Under the Rules tab, there are two sections: 'Edit rules' and 'Monitor rules'. The 'Edit rules' section shows a timestamped log entry from 'Today • 9:51 AM' and another from 'Today • 9:48 AM'. Below the log is a code editor containing the following security rules:

```
1 rules_version = '2';
2 service cloud.firestore {
3     match /databases/{database}/documents {
4         match /{document=**} {
5             allow read, write;
6         }
7     }
8 }
```

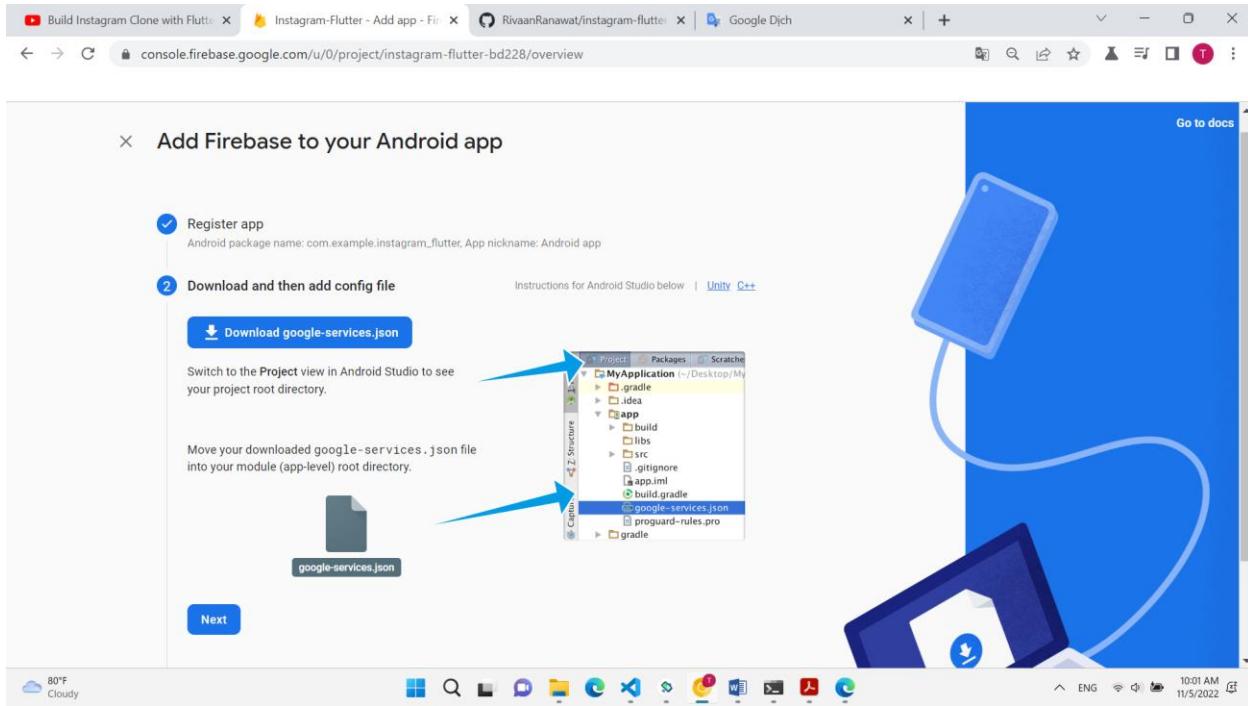
Bước 14. Cấu hình Firebase sử dụng cho android

The screenshot shows the Firebase Project Overview page for the 'Instagram-Flutter' project. The sidebar on the left includes links for Project Overview, Authentication, Firestore Database, and other Firebase products like Build, Release & Monitor, Analytics, Engage, and All products. The main content area features a large blue banner with the text 'Instagram-Flutter' and 'Spark plan'. It encourages users to 'Get started by adding Firebase to your app' and shows icons for Android, iOS+, and other platforms. Below the banner, it says 'Add an app to get started' and 'Store and sync app data in milliseconds'. At the bottom, there's a footer with system status icons.

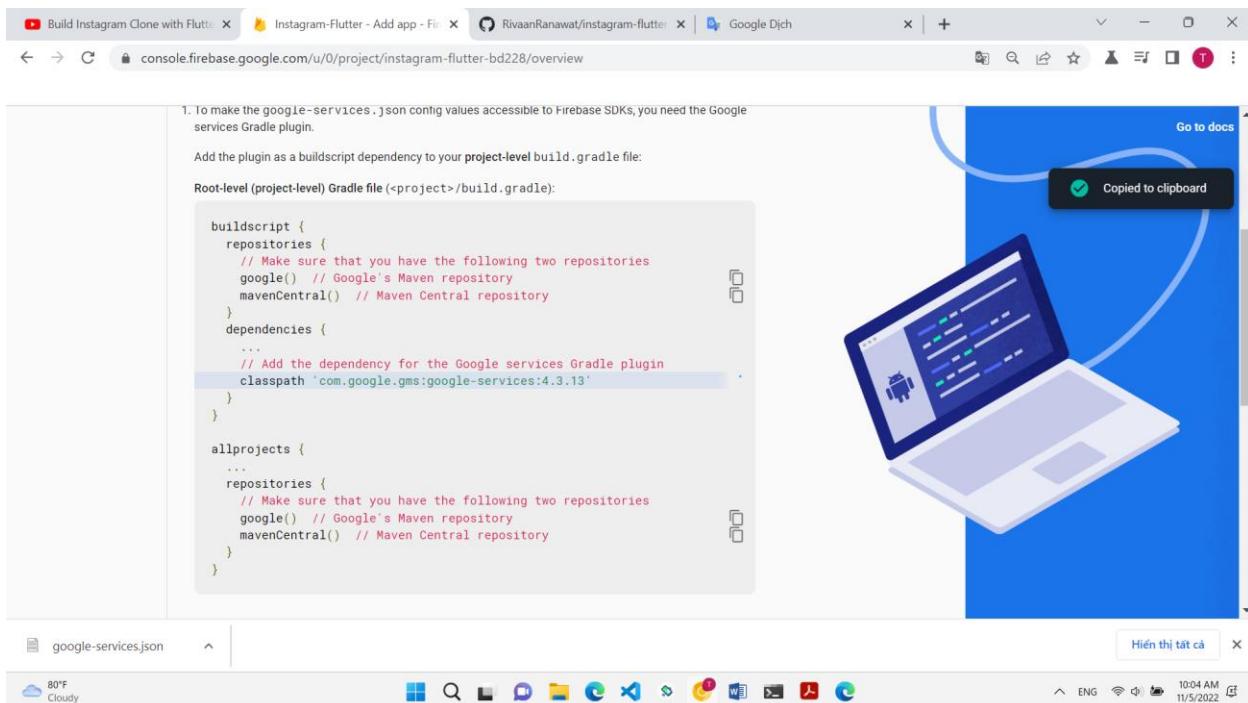
Vào tập tin sau để lấy thông tin

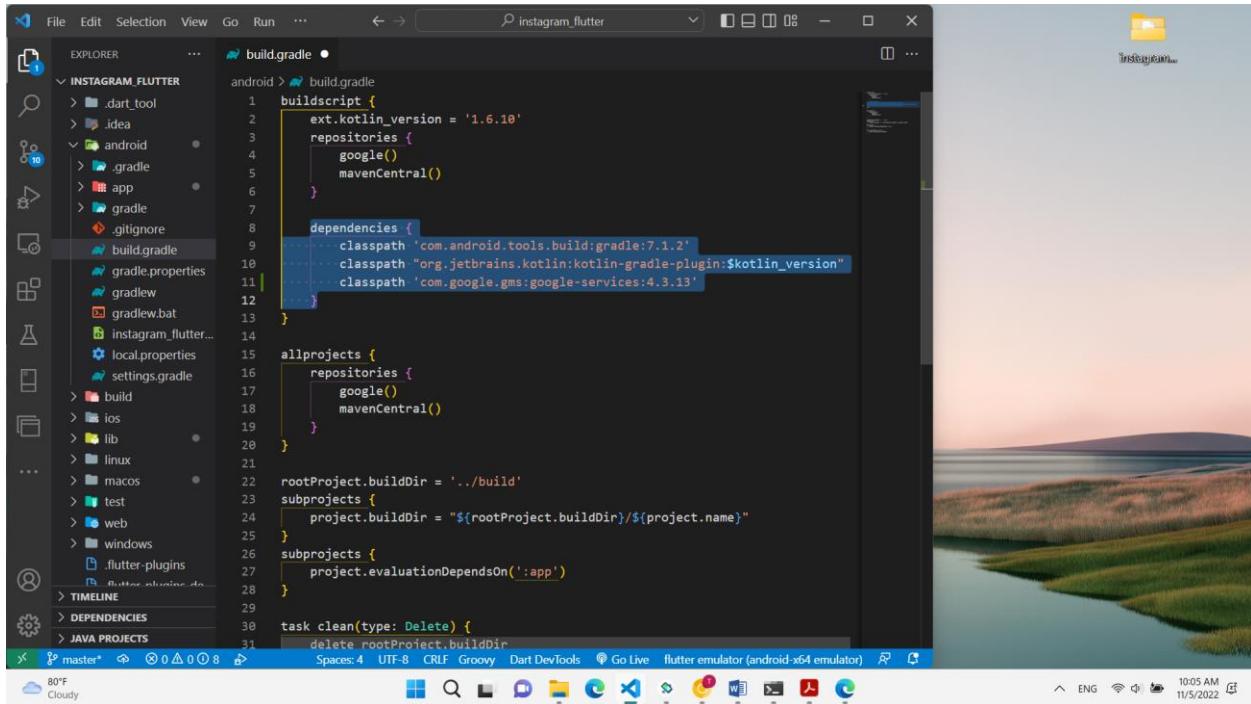
```
build.gradle
32 android > app > build.gradle
33     compileOptions {
34         sourceCompatibility JavaVersion.VERSION_1_8
35         targetCompatibility JavaVersion.VERSION_1_8
36     }
37
38     kotlinOptions {
39         jvmTarget = '1.8'
40     }
41
42     sourceSets {
43         main.java.srcDirs += 'src/main/kotlin'
44     }
45
46     defaultConfig {
47         // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id)
48         applicationId "com.example.instagram_flutter"
49         // You can update the following values to match your application needs.
50         // For more information, see: https://docs.flutter.dev/deployment/android
51         minSdkVersion flutter.minSdkVersion
52         targetSdkVersion flutter.targetSdkVersion
53         versionCode flutterVersionCode.toInt()
54         versionName flutterVersionName
55     }
56
57     buildTypes {
58         release {
59             // TODO: Add your own signing config for the release build.
60             // Signing with the debug keys for now, so 'flutter run --release'.
61             signingConfig signingConfigs.debug
62         }
63     }
64 }
```

Bước 15. Download và copy vào thư mục như hướng dẫn



Copy dòng bên dưới và lưu vào





```
buildscript {
    ext.kotlin_version = '1.6.10'
    repositories {
        google()
        mavenCentral()
    }

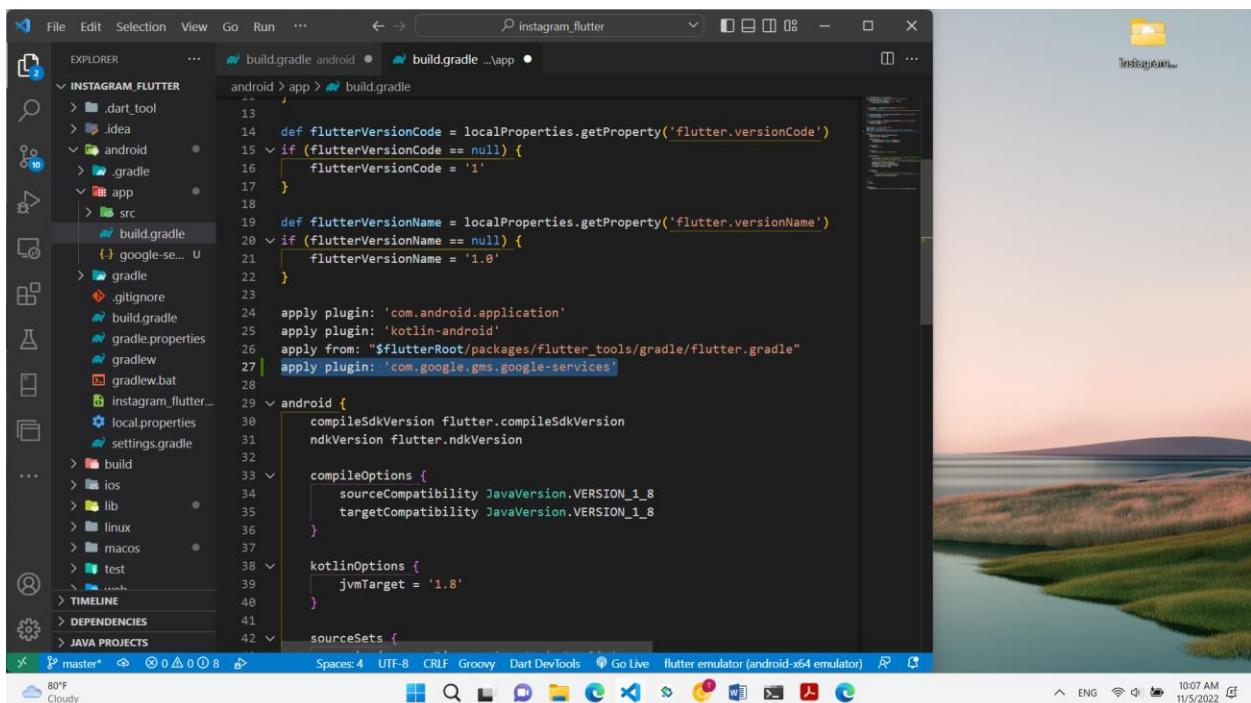
    dependencies {
        classpath 'com.android.tools.build:gradle:7.1.2'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
        classpath 'com.google.gms:google-services:4.3.13'
    }
}

allprojects {
    repositories {
        google()
        mavenCentral()
    }
}

rootProject.buildDir = '../build'
subprojects {
    project.buildDir = "${rootProject.buildDir}/${project.name}"
}
subprojects {
    project.evaluationDependsOn(':app')
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Thêm dòng sau



```
def flutterVersionCode = localProperties.getProperty('flutter.versionCode')
if (flutterVersionCode == null) {
    flutterVersionCode = '1'
}

def flutterVersionName = localProperties.getProperty('flutter.versionName')
if (flutterVersionName == null) {
    flutterVersionName = '1.0'
}

apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply from: "$flutterRoot/packages/flutter_tools/gradle/flutter.gradle"
apply plugin: 'com.google.gms.google-services'

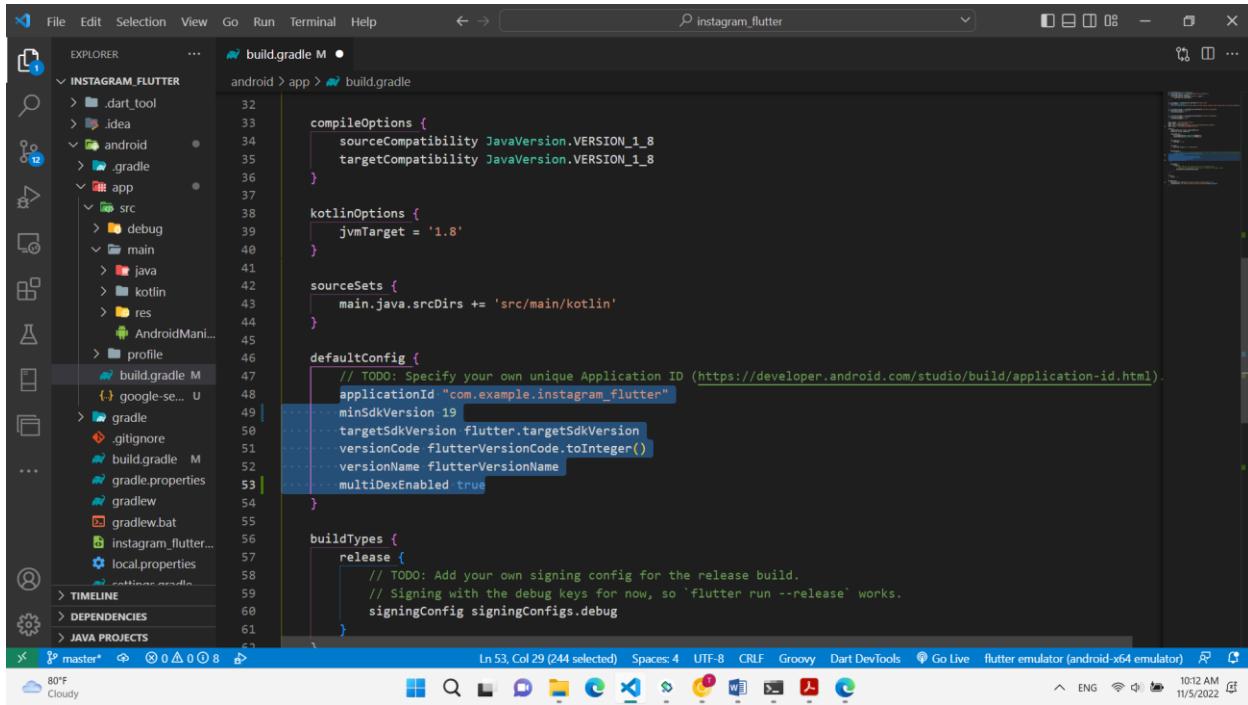
android {
    compileSdkVersion flutter.compileSdkVersion
    ndkVersion flutter.ndkVersion

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

    kotlinOptions {
        jvmTarget = '1.8'
    }

    sourceSets {
```

Chỉnh nội dung tập tin



```
build.gradle M
...
compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
}

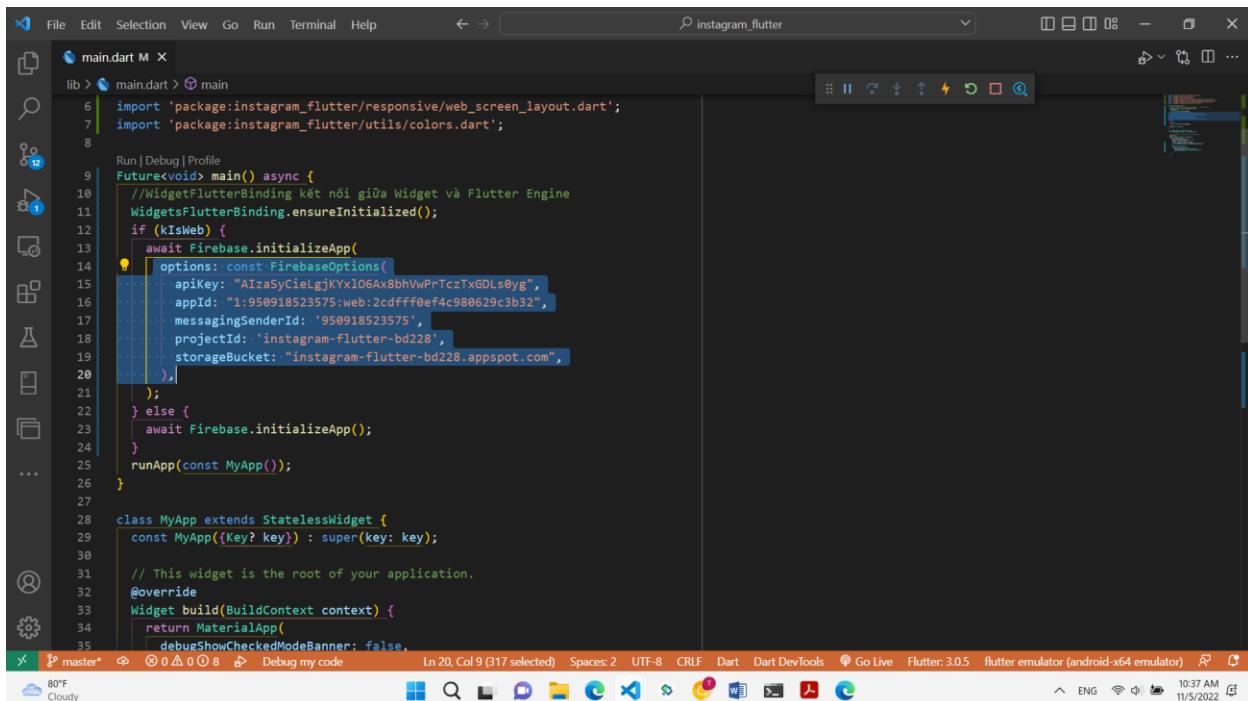
kotlinOptions {
    jvmTarget = '1.8'
}

sourceSets {
    main.java.srcDirs += 'src/main/kotlin'
}

defaultConfig {
    // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html)
    applicationId "com.example.instagram_flutter"
    minSdkVersion 19
    targetSdkVersion flutter.targetSdkVersion
    versionCode flutterVersionCode.toInt()
    versionName flutterVersionName
    multiDexEnabled true
}

buildTypes {
    release {
        // TODO: Add your own signing config for the release build.
        // Signing with the debug keys for now, so 'flutter run --release' works.
        signingConfig signingConfigs.debug
    }
}
```

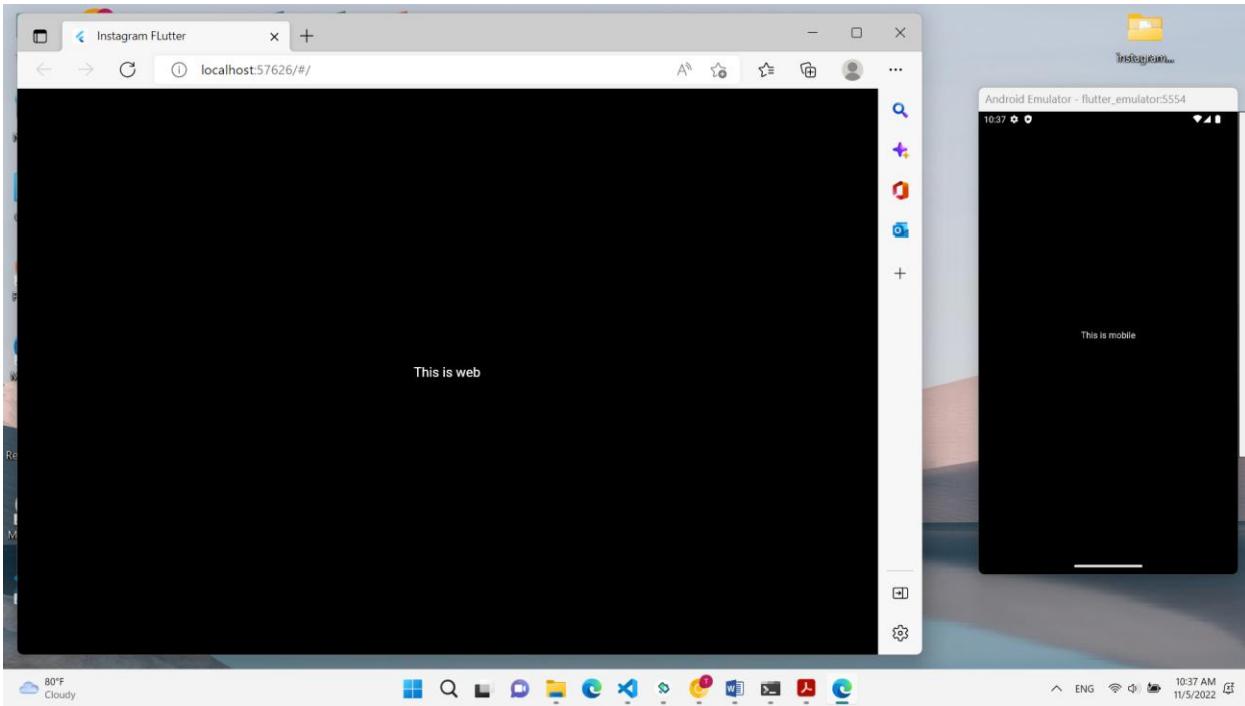
Cấu hình các thông số để kết nối web



```
main.dart M
...
Future<void> main() async {
    WidgetsFlutterBinding.ensureInitialized();
    if (kIsWeb) {
        await Firebase.initializeApp(
            options: const FirebaseOptions(
                apiKey: "AIzaSyCielGjkYXl06AxbbhVwPrTczTxGDls8yg",
                appId: "1:950918523575:web:2cdfff0ef4c980629c3b32",
                messagingSenderId: '950918523575',
                projectId: 'instagram-flutter-bd228',
                storageBucket: "instagram-flutter-bd228.appspot.com",
            ),
        );
    } else {
        await Firebase.initializeApp();
    }
    runApp(const MyApp());
}

class MyApp extends StatelessWidget {
    const MyApp({Key? key}) : super(key: key);

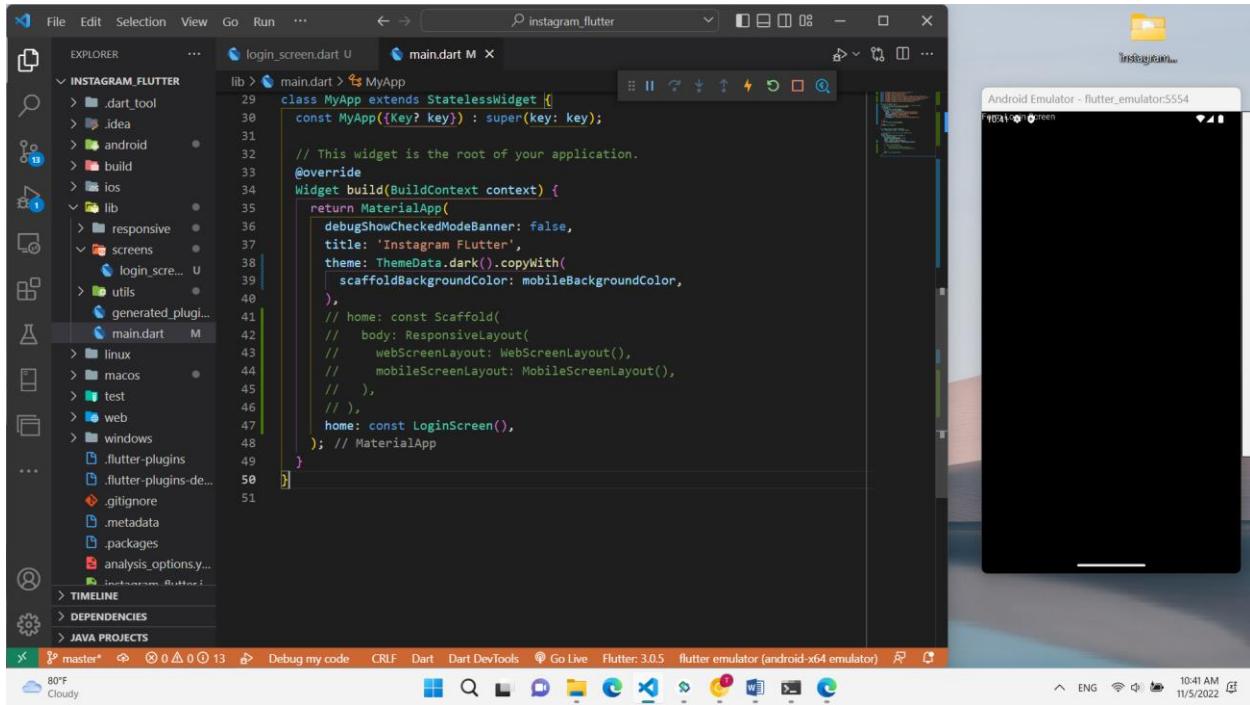
    // This widget is the root of your application.
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            debugShowCheckedModeBanner: false,
        );
    }
}
```



Bước 16. Tạo trang login

```
lib> screens > login_screen.dart > LoginScreen
1 import 'package:flutter/material.dart';
2
3 class LoginScreen extends StatefulWidget {
4   const LoginScreen({Key? key}) : super(key: key);
5
6   @override
7   State<LoginScreen> createState() => _LoginScreenState();
8 }
9
10 class _LoginScreenState extends State<LoginScreen> {
11   @override
12   Widget build(BuildContext context) {
13
14   }
15 }
16
```

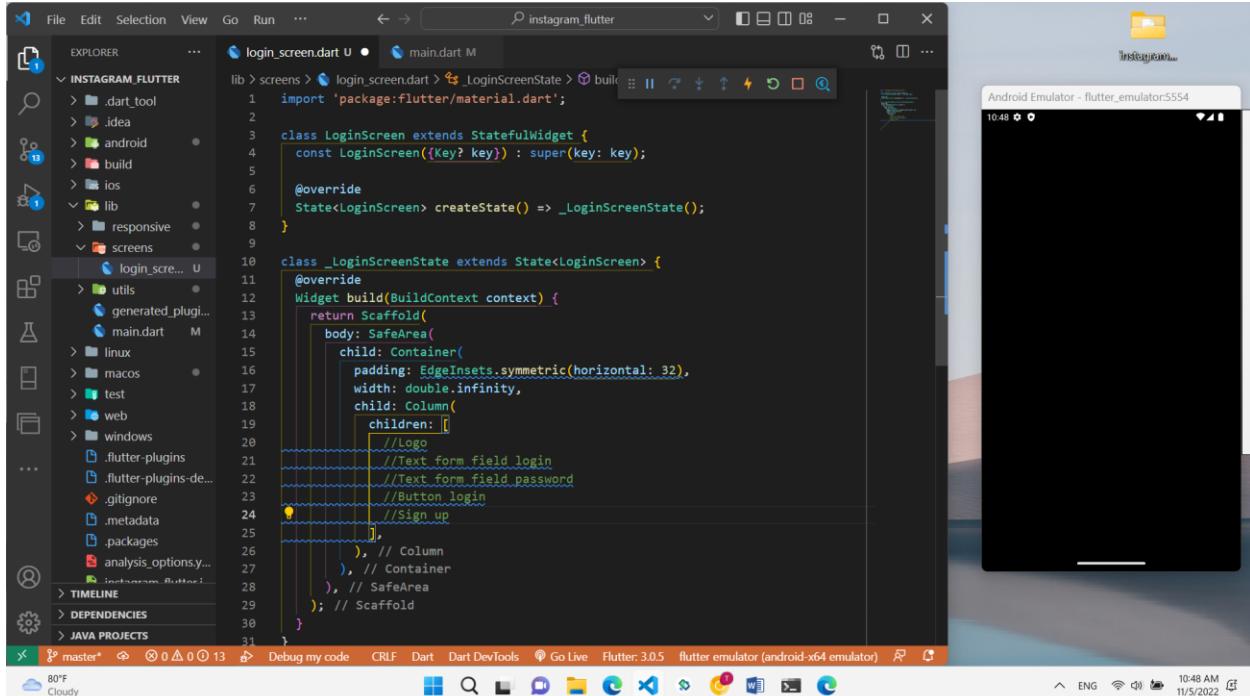
Test login in chạy tốt



```
lib > main.dart > MyApp
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Instagram Flutter',
      theme: ThemeData.dark().copyWith(
        scaffoldBackgroundColor: mobileBackgroundColor,
      ),
      // home: const Scaffold(
      //   body: ResponsiveLayout(
      //     webScreenLayout: WebScreenLayout(),
      //     mobileScreenLayout: MobileScreenLayout(),
      //   ),
      //   home: const LoginScreen(),
      // ); // MaterialApp
```

Ý tưởng xây dựng trang password chia thành các thành phần khác nhau nằm trong cột4



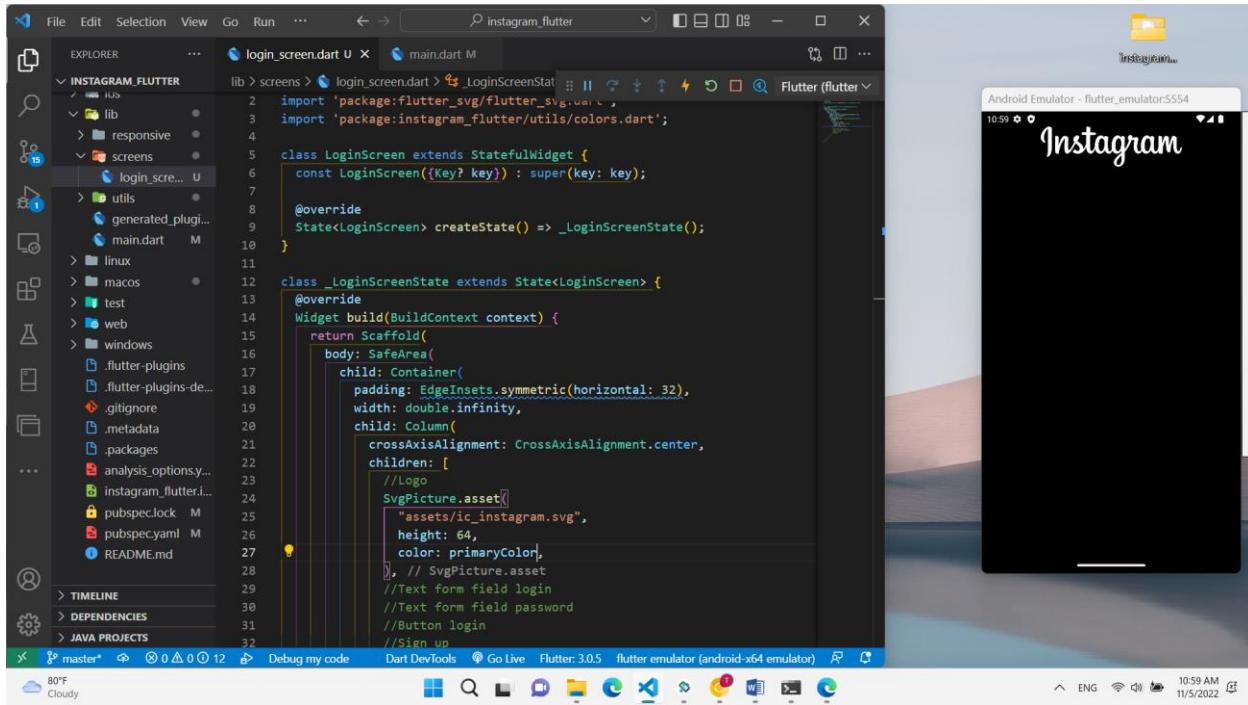
```
lib > screens > login_screen.dart > _LoginScreenState > build
import 'package:flutter/material.dart';

class LoginScreen extends StatefulWidget {
  const LoginScreen({Key? key}) : super(key: key);

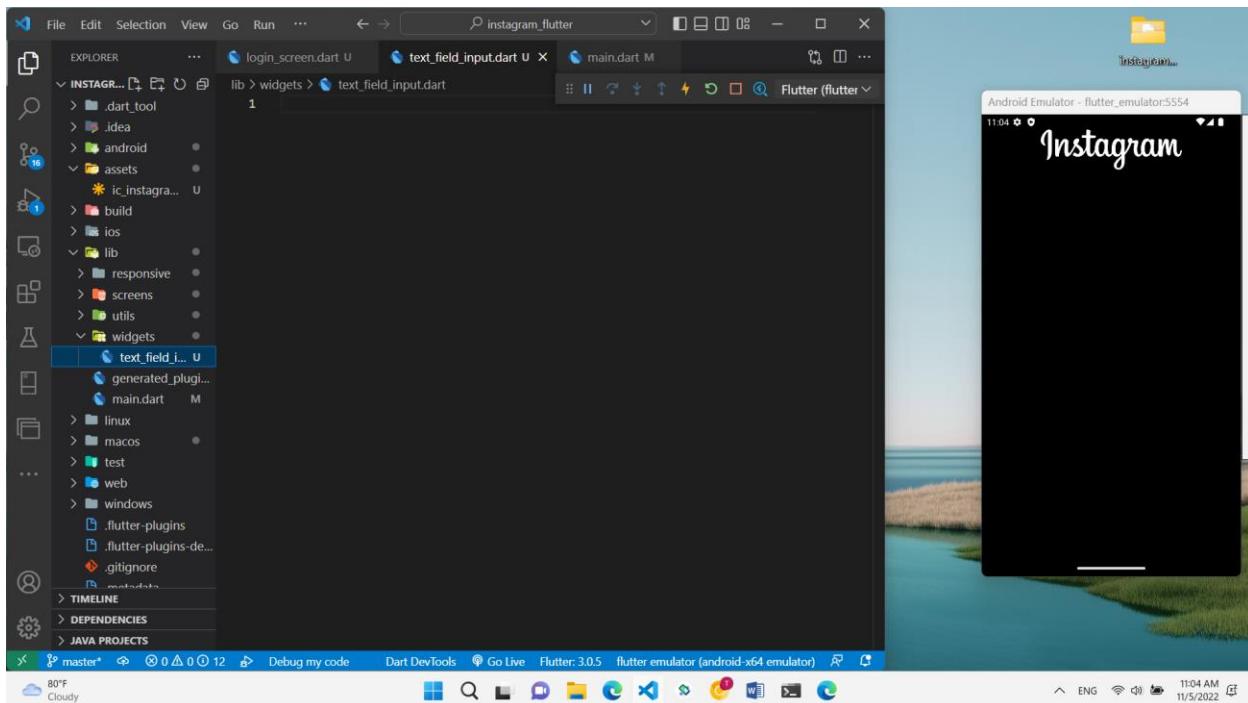
  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Container(
          padding: EdgeInsets.symmetric(horizontal: 32),
          width: double.infinity,
          child: Column(
            children: [
              //Logo
              //Text form field login
              //Text form field password
              //Button login
              //Sign up
            ],
          ), // Column
        ), // Container
      ), // SafeArea
    ); // Scaffold
}
```

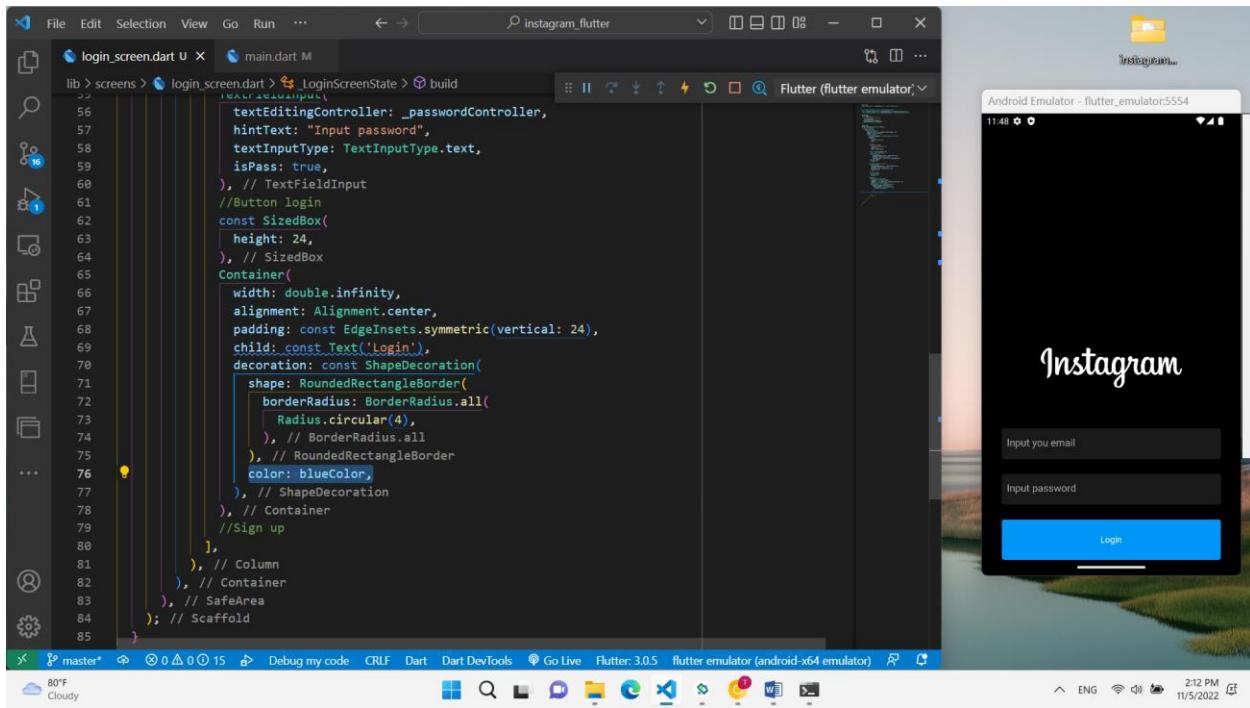
Tạo logo



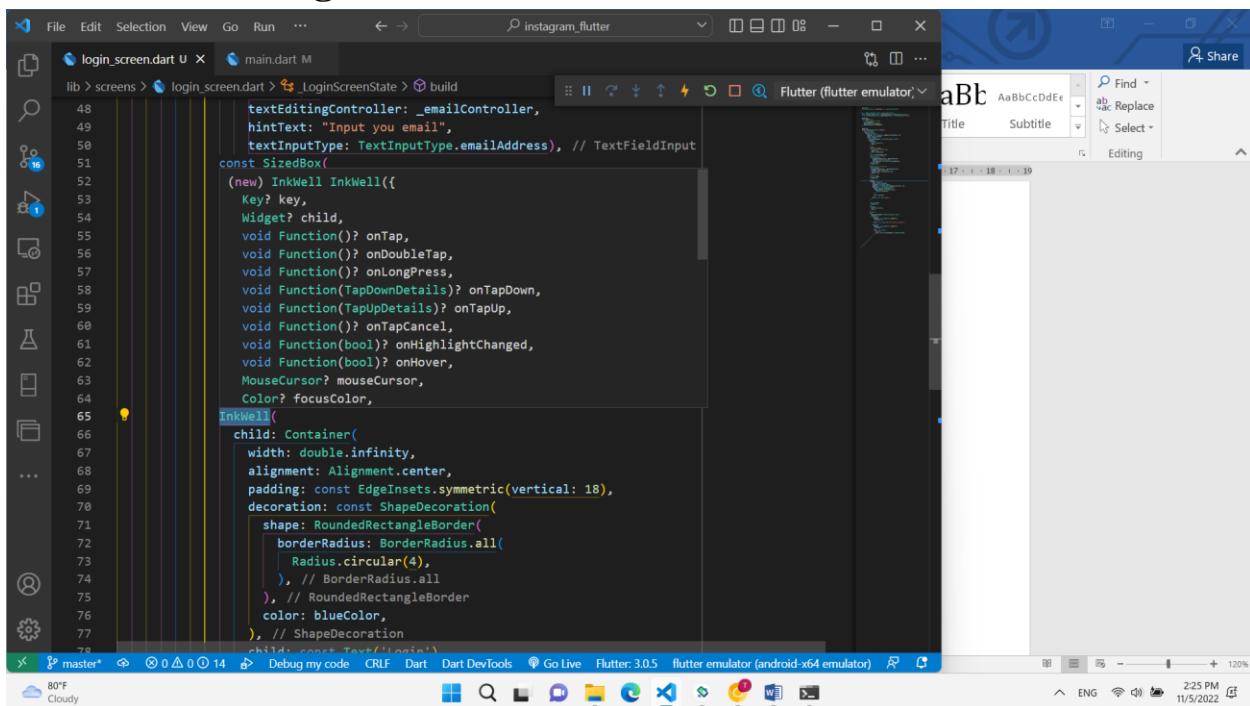
Tạo text_field_input.dart



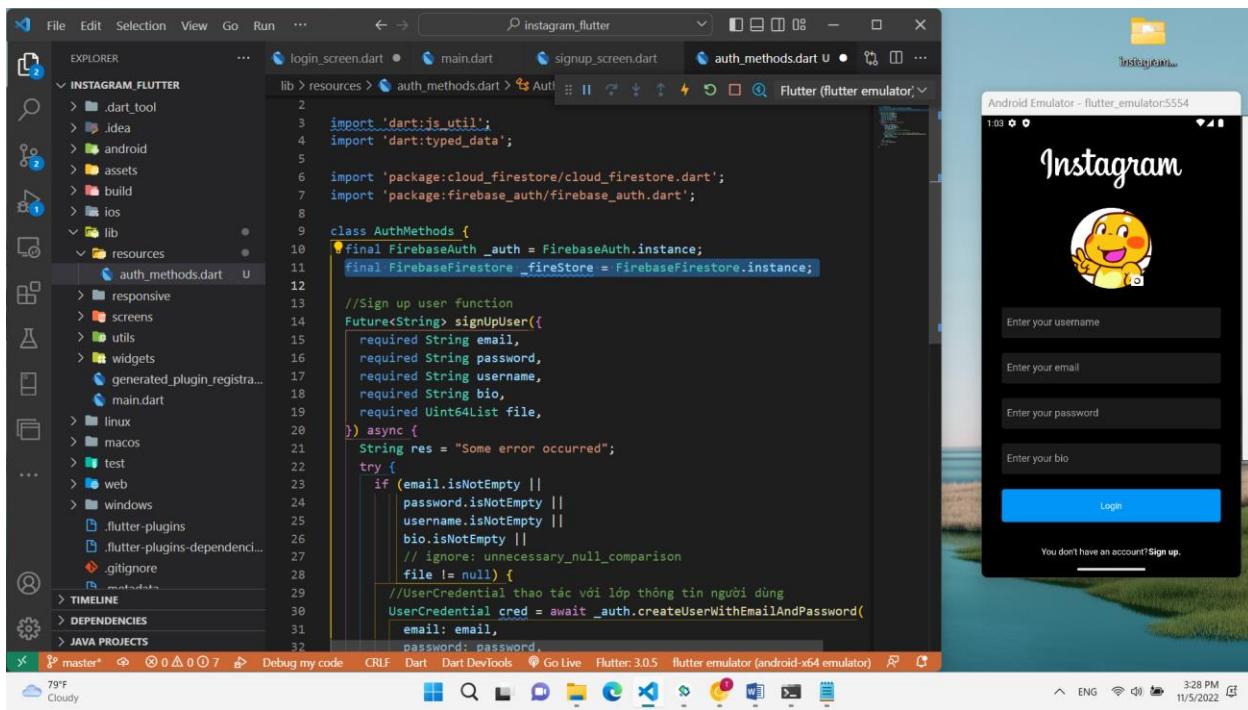
Bước 17. Thêm màu sắc cho nút login



Bước 18. Bao login bởi

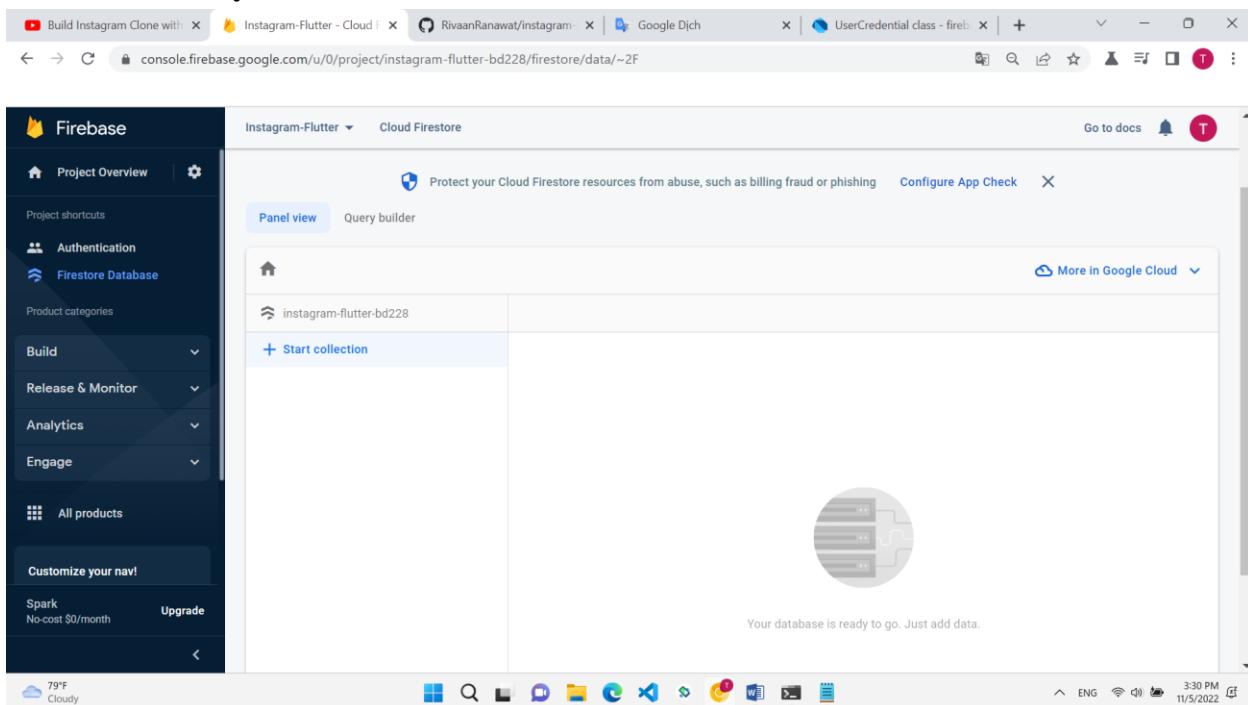


Bước 19. Xây dựng trang Sign up



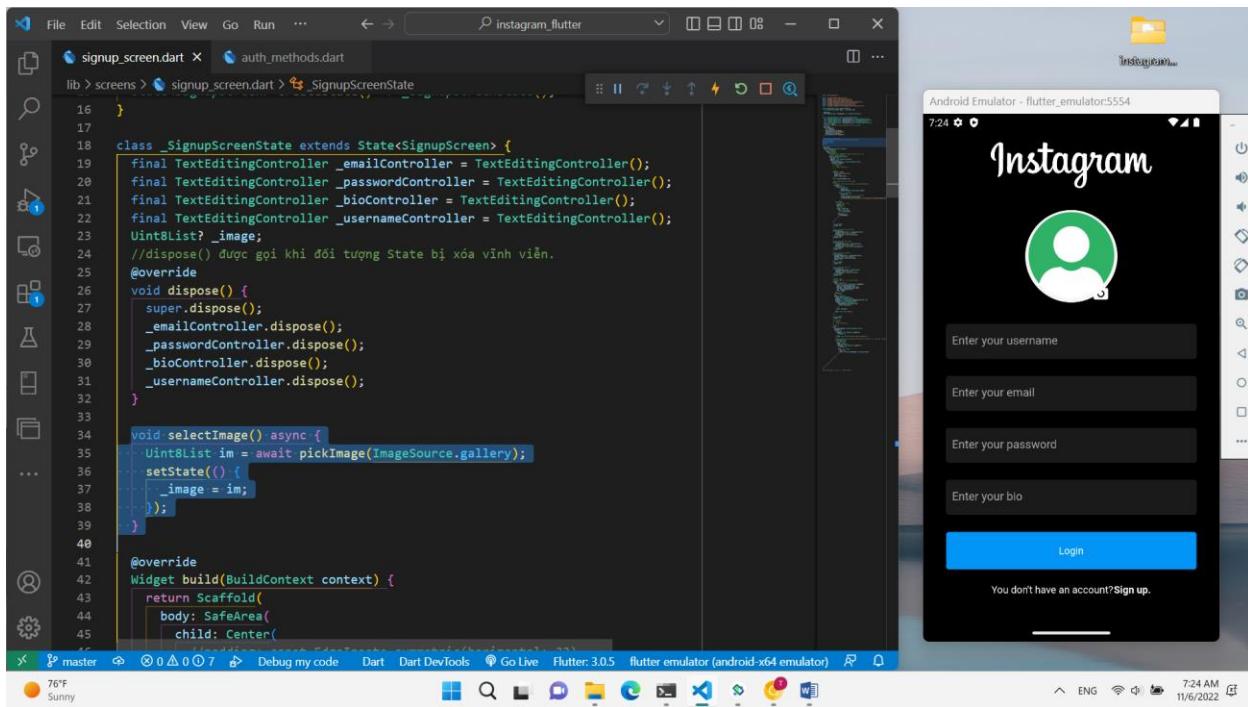
Thêm FirebaseFirestore để thêm users vào database

Bước 20. Tạo collection mới tên là users



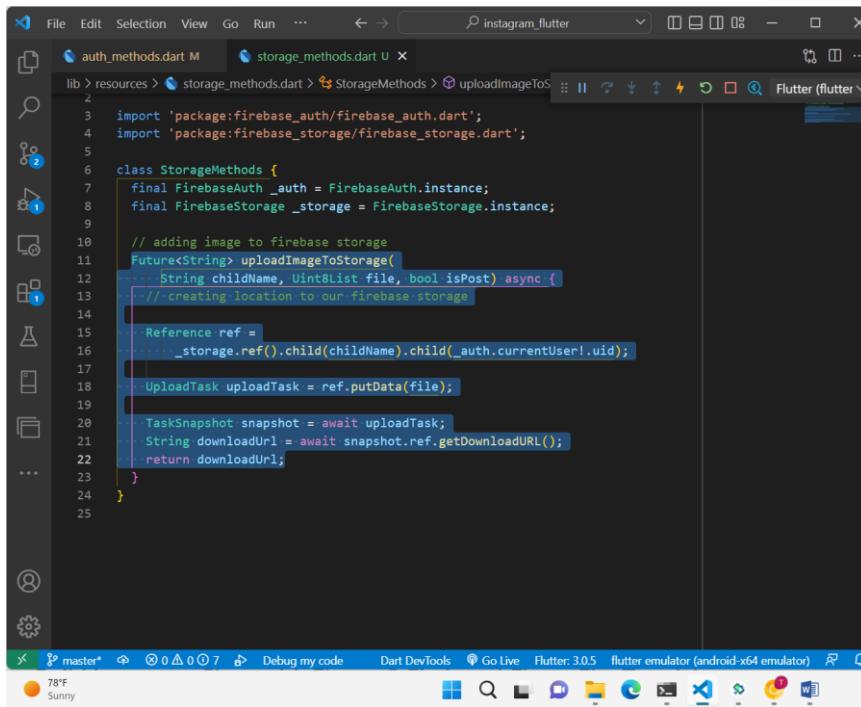
Bước 21. Viết hàm để load hình ảnh

Trong lib/utils tạo utils.dart để chứa các hàm

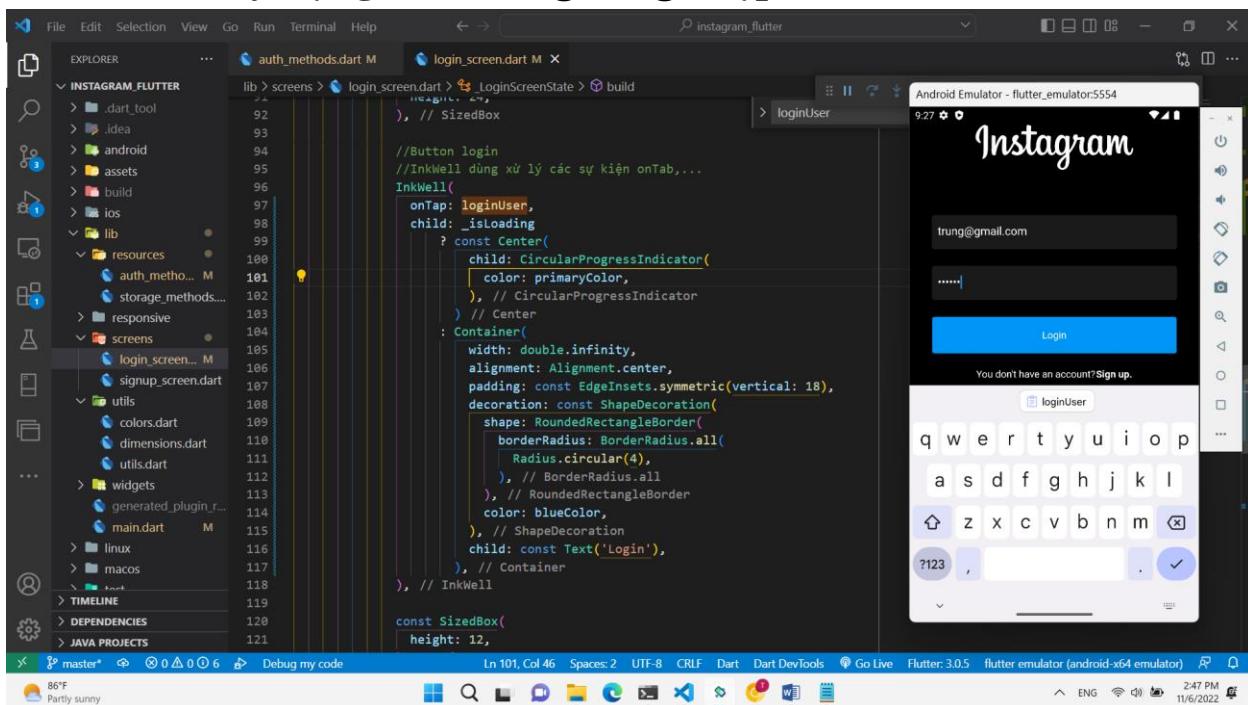


Bước 22. Hàm để upload và download ảnh

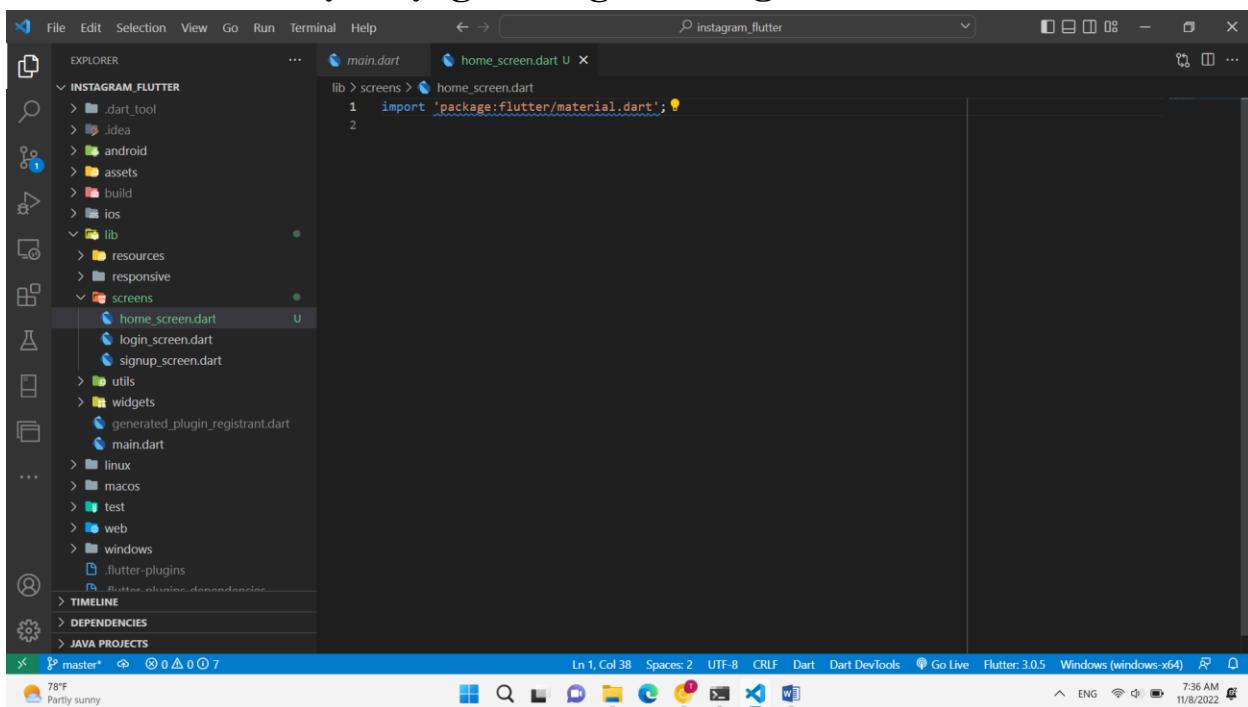
***Lưu ý chỉnh sửa các rule để được phép upload và download

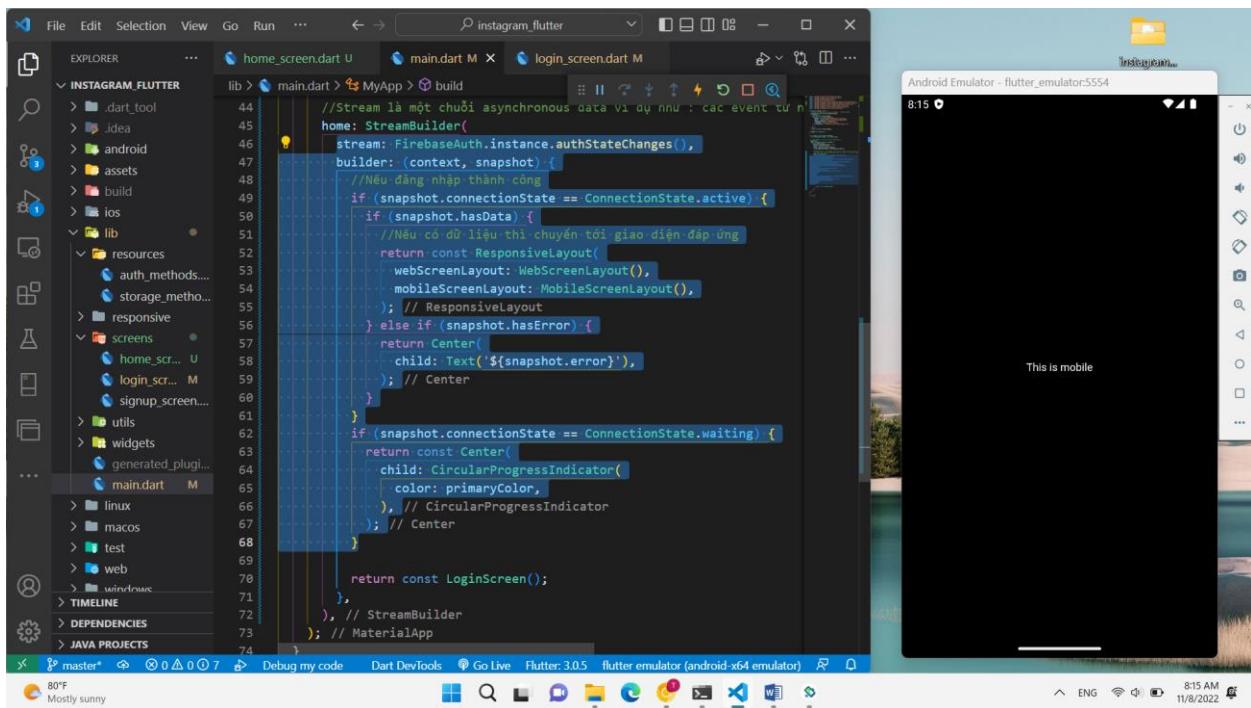


Bước 23. Xây dựng chức năng đăng nhập



Bước 24. Xác thực trạng thái người dùng

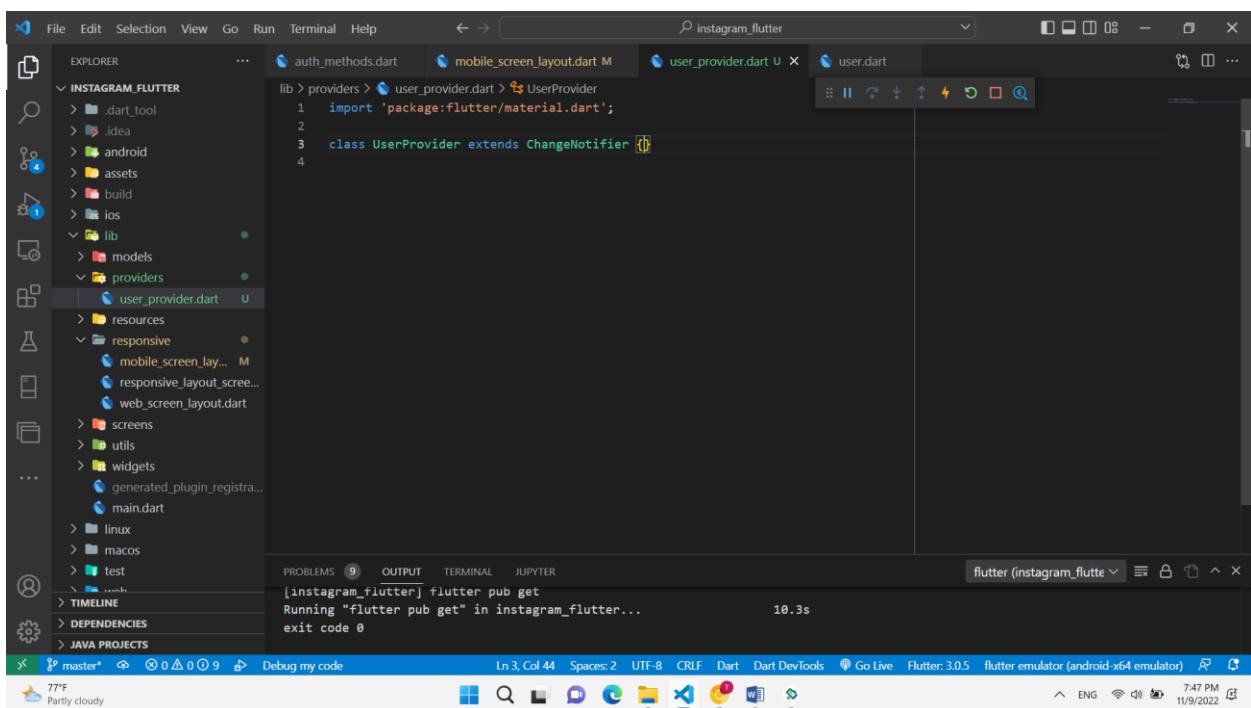




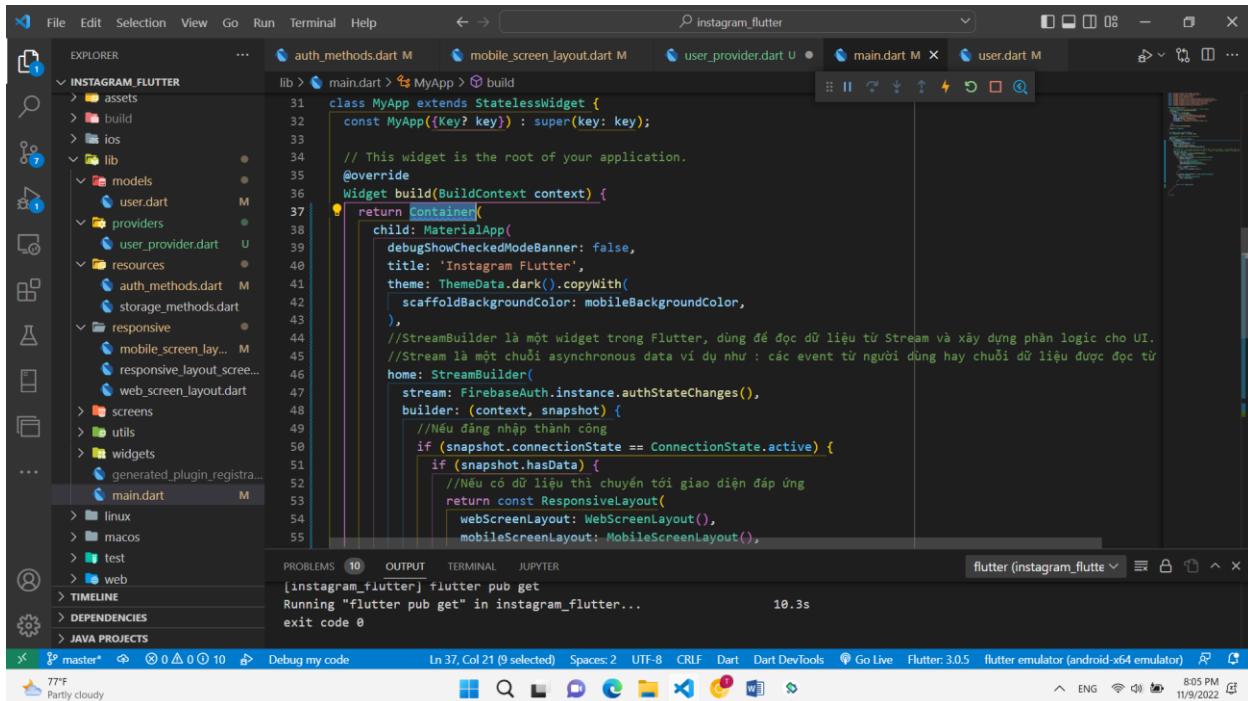
Bước 25. Lập mô hình dữ liệu người dùng

Bước 26. Quản lý trạng thái dữ liệu người dùng

Sử dụng provider để quản lý => tải provider về



Bước 27. Chuyển đổi main.dart



The screenshot shows the VS Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** instagram_flutter
- Explorer:** Shows the project structure for 'INSTAGRAM_FLUTTER'. The 'lib' folder contains 'main.dart', 'auth_methods.dart', 'mobile_screen_layout.dart', 'user_provider.dart', and 'user.dart'. Other folders like 'ios', 'lib', 'models', 'providers', 'resources', 'responsive', 'screens', 'utils', 'widgets', and 'generated_plugin_registration.dart' are also listed.
- Main.dart Content:**

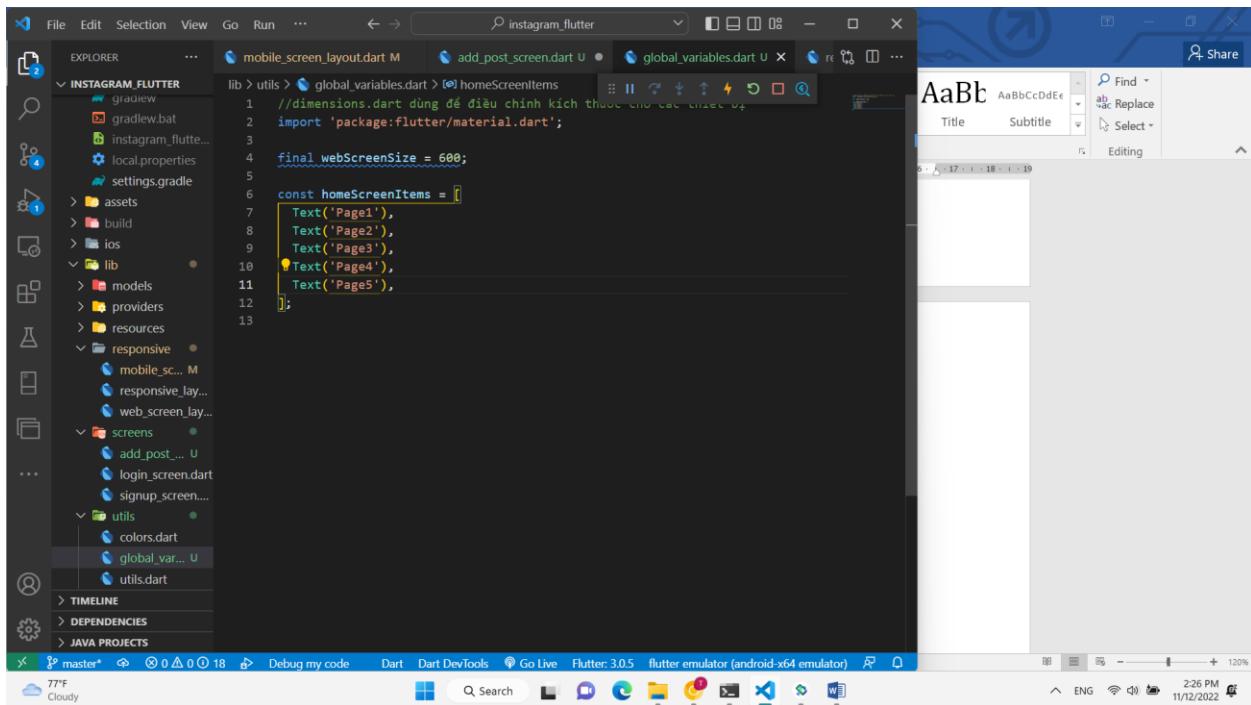
```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return Container(
      child: MaterialApp(
        debugShowCheckedModeBanner: false,
        title: 'Instagram Flutter',
        theme: ThemeData.dark().copyWith(
          scaffoldBackgroundColor: mobileBackgroundColor,
        ),
        //StreamBuilder là một widget trong Flutter, dùng để đọc dữ liệu từ Stream và xây dựng phần logic cho UI.
        //Stream là một chuỗi asynchronous data ví dụ như : các event từ người dùng hay chuỗi dữ liệu được đọc từ
        home: StreamBuilder(
          stream: FirebaseAuth.instance.authStateChanges(),
          builder: (context, snapshot) {
            //Nếu đang nhập thành công
            if (snapshot.connectionState == ConnectionState.active) {
              if (snapshot.hasData) {
                //Nếu có dữ liệu thì chuyển tới giao diện đáp ứng
                return const ResponsiveLayout(
                  webScreenLayout: WebScreenLayout(),
                  mobileScreenLayout: MobileScreenLayout(),
                );
              }
            }
          }
        )
      );
  }
}
```
- Terminal:** Shows the command: flutter pub get. It indicates the process is running and took 10.3s.
- Bottom Status Bar:** Includes icons for file operations, a progress bar, and system status like battery and network.

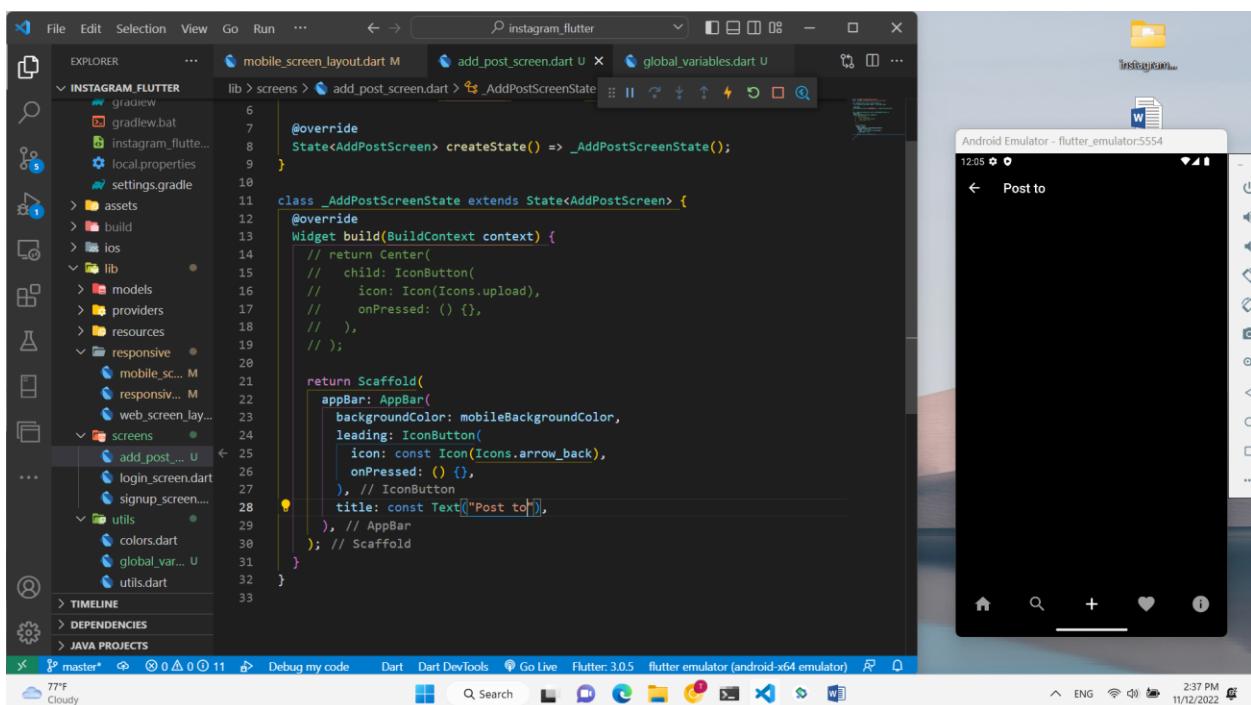
Bước 28. Chuyển đổi responsive layout thành stafulWidget

Bước 29. Xây dựng trang add post

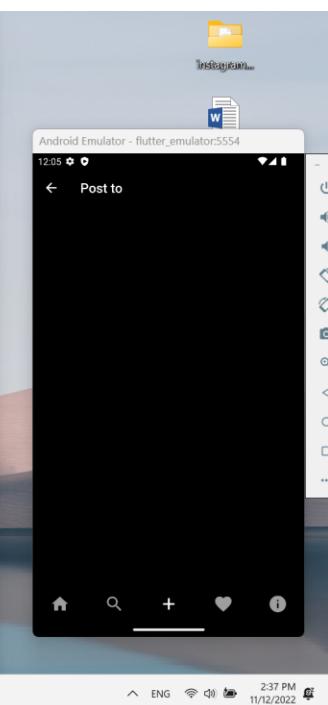
Đổi tên tập tin dis thành global_variable.dart

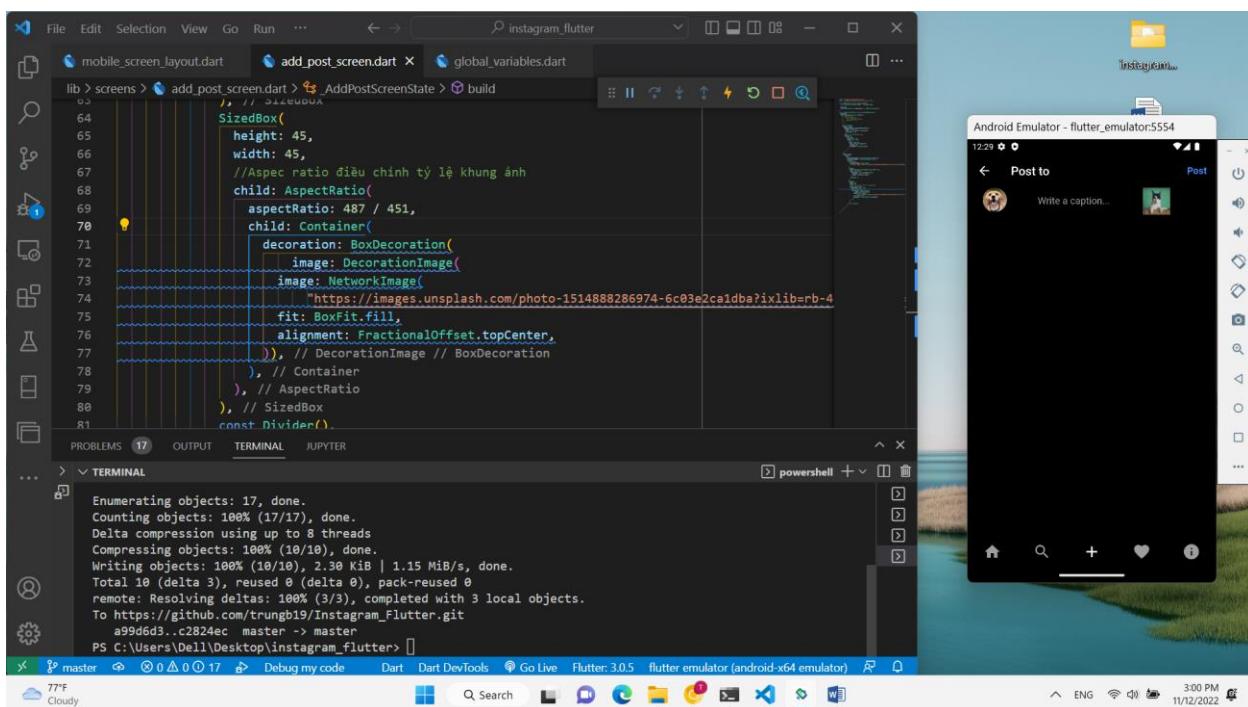
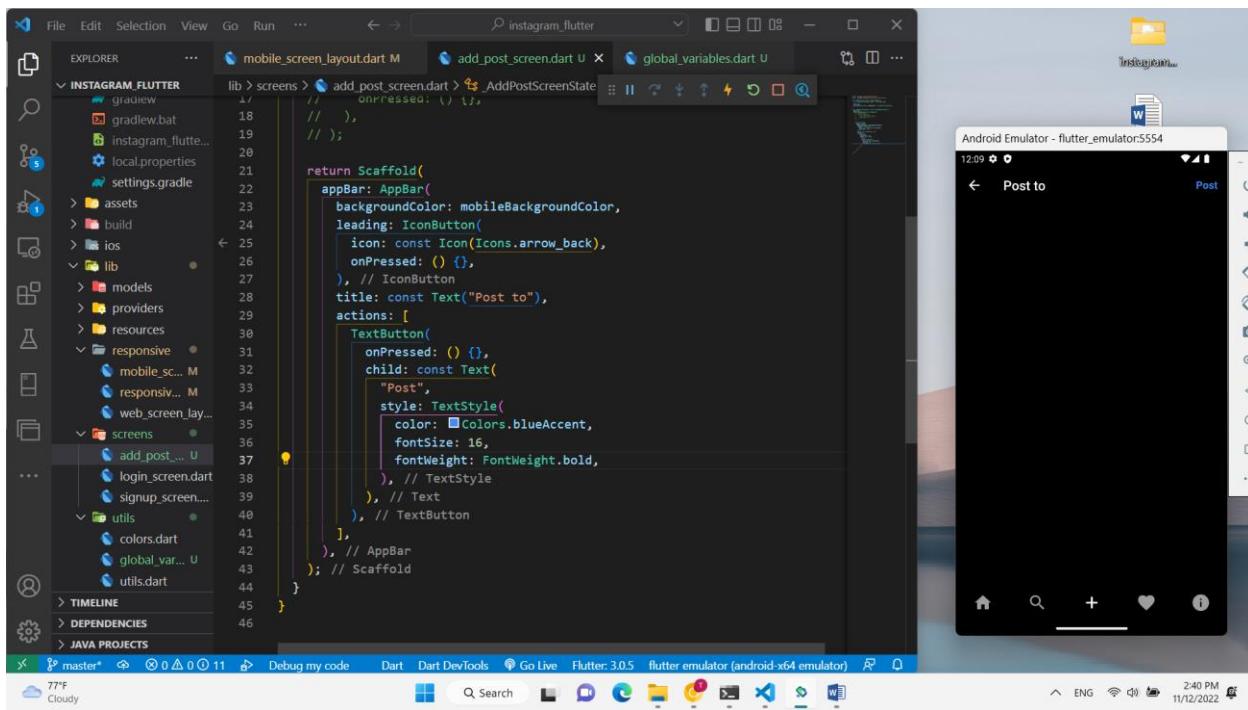


```
lib > utils > global_variables.dart > homeScreenItems
1 //dimensions.dart dùng để điều chỉnh kích thước cho các thiết bị
2 import 'package:flutter/material.dart';
3
4 final webScreenSize = 600;
5
6 const homeScreenItems = [
7   Text('Page1'),
8   Text('Page2'),
9   Text('Page3'),
10  Text('Page4'),
11  Text('Page5')
12];
```



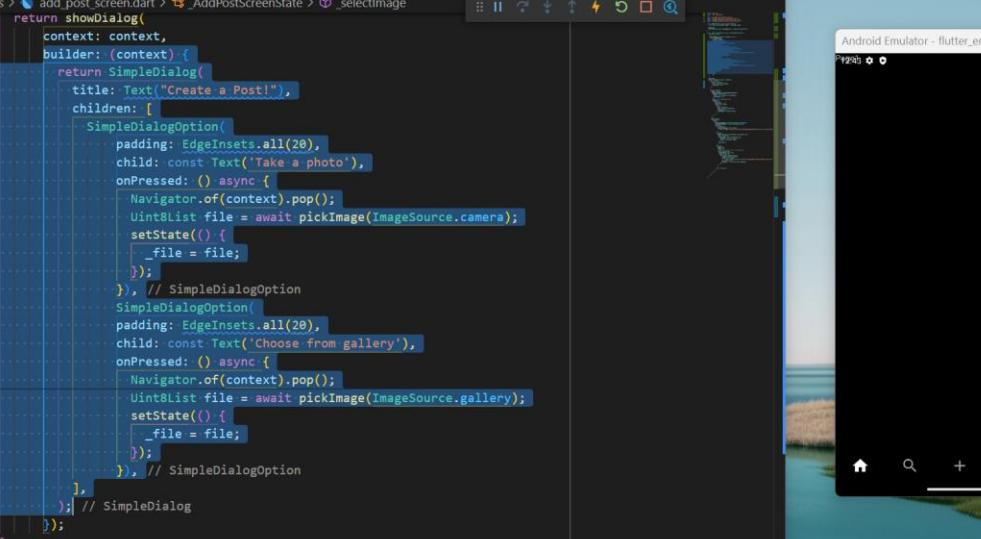
```
lib > screens > add_post_screen.dart > _AddPostScreenState
6
7 @override
8 State<AddPostScreen> createState() => _AddPostScreenState();
9 }
10
11 class _AddPostScreenState extends State<AddPostScreen> {
12   @override
13   Widget build(BuildContext context) {
14     // return Center(
15     //   child: IconButton(
16     //     icon: Icon(Icons.upload),
17     //     onPressed: () {},
18     //   ),
19     // );
20
21     return Scaffold(
22       appBar: AppBar(
23         backgroundColor: mobileBackgroundColor,
24         leading: IconButton(
25           icon: const Icon(Icons.arrow_back),
26           onPressed: () {},
27         ), // IconButton
28         title: const Text("Post to"),
29       ), // AppBar
30     ); // Scaffold
31   }
32 }
```





Bước 30. Select image

Viết hàm _selectimage



A screenshot of a Flutter development environment in VS Code. The code editor shows a Dart file named `add_post_screen.dart` with the following code:

```
lib > screens > add_post_screen.dart M > _AddPostScreenState > _selectImage
20   return showDialog(
21     context: context,
22     builder: (context) {
23       return SimpleDialog(
24         title: Text("Create a Post"),
25         children: [
26           SimpleDialogOption(
27             padding: EdgeInsets.all(20),
28             child: const Text('Take a photo'),
29             onPressed: () async {
30               Navigator.of(context).pop();
31               Uint8List file = await pickImage(ImageSource.camera);
32               setState(() {
33                 _file = file;
34               });
35             },
36           ), // SimpleDialogOption
37           SimpleDialogOption(
38             padding: EdgeInsets.all(20),
39             child: const Text('Choose from gallery'),
40             onPressed: () async {
41               Navigator.of(context).pop();
42               Uint8List file = await pickImage(ImageSource.gallery);
43               setState(() {
44                 _file = file;
45               });
46             },
47           ), // SimpleDialogOption
48         ],
49       ); // SimpleDialog
50     );
51   }
```

The right side of the screen shows the Android emulator displaying a landscape photo of a beach and water. The top status bar indicates the Instagram logo.

Hiệu chỉnh builder

The screenshot shows a development environment for a Flutter application. On the left, the code editor displays `mobile_screen_layout.dart` with several UI components like `IconButtons` and `TextButtons`. The right side shows an Android emulator window titled "Android Emulator - flutter_emulator:5554" displaying a landscape view of a lake and reeds.

```
51 @override
52 Widget build(BuildContext context) {
53   return _file == null
54     ? Center(
55       child: IconButton(
56         icon: Icon(Icons.upload),
57         onPressed: () {},
58       ), // IconButton
59     ) // Center
60     : Scaffold(
61       appBar: AppBar(
62         backgroundColor: mobileBackgroundColor,
63         leading: IconButton(
64           icon: const Icon(Icons.arrow_back),
65           onPressed: () {},
66         ), // IconButton
67         title: const Text("Post to"),
68         actions: [
69           TextButton(
70             onPressed: () {},
71             child: const Text(
72               "Post",
73               style: TextStyle(
74                 color: Colors.blueAccent,
75                 fontSize: 16,
76                 fontWeight: FontWeight.bold,
77               ), // TextStyle
78             ), // Text
79           ), // TextButton
80         ],
81       ), // AppBar
82     );
83 }
```

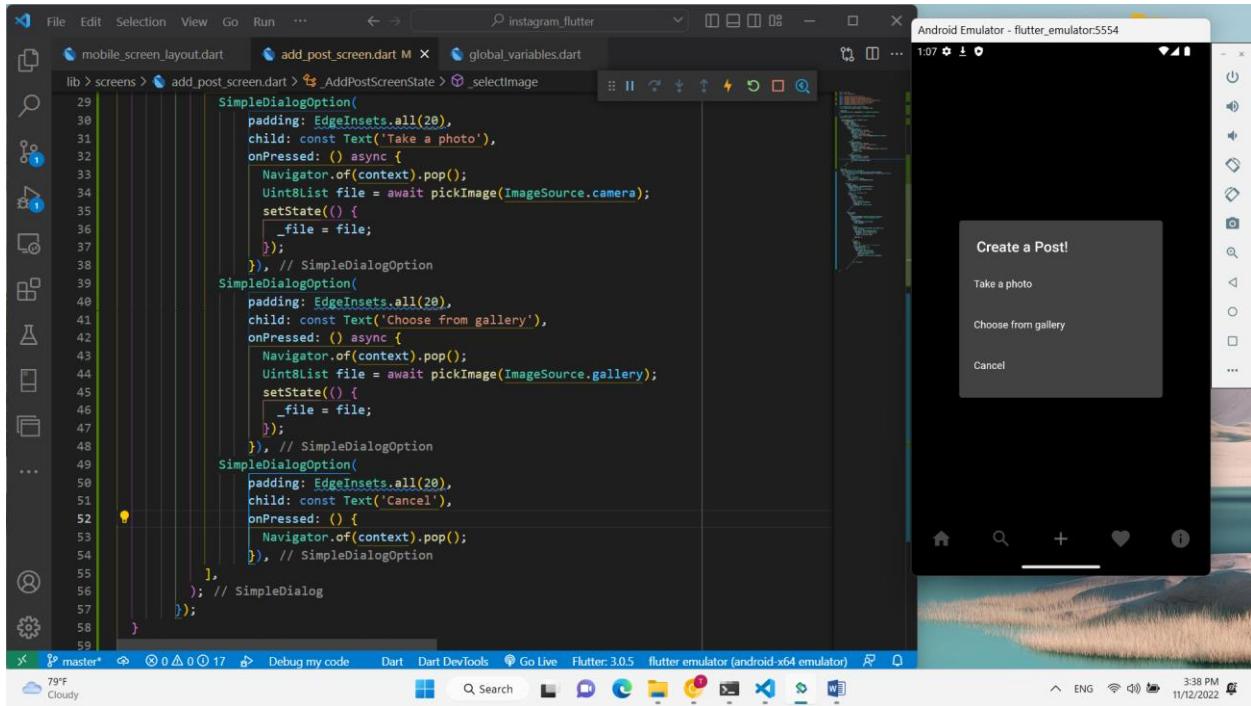
A screenshot of the Visual Studio Code interface. The left pane shows the code for `add_post_screen.dart`. The right pane displays the `flutter_emulator` showing the Instagram app's "Create a Post!" screen. The screen has a title "Create a Post!", two buttons ("Take a photo" and "Choose from gallery"), and a large image placeholder at the bottom.

```
lib > screens > add_post_screen.dart > _AddPostScreenState > build
  ...
  Navigator.of(context).pop();
  Uint8List file = await pickImage(ImageSource.gallery);
  setState(() {
    _file = file;
  });
}, // SimpleDialogOption
); // SimpleDialog
);
}
}

@Override
Widget build(BuildContext context) {
  return _file == null
    ? Center(
      child: IconButton(
        icon: Icon(Icons.upload),
        onPressed: () => _selectImage(context),
      ), // IconButton
    ) // Center
    : Scaffold(
      appBar: AppBar(
        backgroundColor: mobileBackgroundColor,
        leading: IconButton(
          icon: const Icon(Icons.arrow_back),
          onPressed: () {},
        ), // IconButton
        title: const Text("Post to"),
        actions: [
          TextButton(
            ...
        ],
      ),
    );
}
```

A screenshot of the Visual Studio Code interface. The left pane shows the code for `add_post_screen.dart`, specifically focusing on the post creation logic. The right pane displays the `flutter_emulator` showing the Instagram app's "Post to" screen. The screen has a back button, a profile picture placeholder, a caption input field, and a "Post" button.

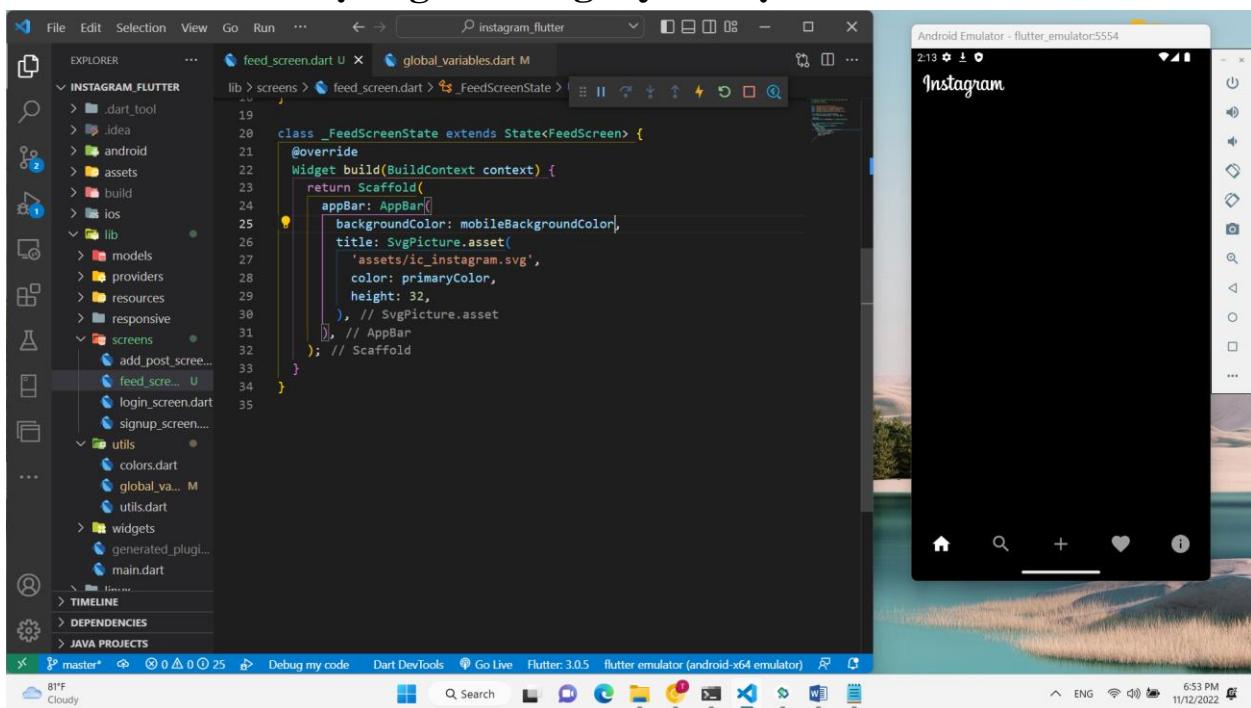
```
lib > screens > add_post_screen.dart > _AddPostScreenState > build
  ...
  border: InputBorder.none,
  ...
  maxLines: 8,
  ), // TextField
), // SizedBox
SizedBox(
  height: 45,
  width: 45,
  // Aspec ratio điều chỉnh tỷ lệ khung ảnh
  child: AspectRatio(
    aspectRatio: 487 / 451,
    child: Container(
      decoration: BoxDecoration(
        image: DecorationImage(
          image: DecorationImage(
            image: MemoryImage(_file),
            fit: BoxFit.fill,
            alignment: FractionalOffset.topCenter,
          ), // DecorationImage // BoxDecoration
        ), // Container
      ), // AspectRatio
    ), // SizedBox
    const Divider(),
  ),
),
], // Row
],
), // Column
); // Scaffold
}
```



Bước 31. Lưu trữ post lên firebasestore

Tải về uuid

Bước 32. Giao diện người dùng lấy dữ liệu từ firebase

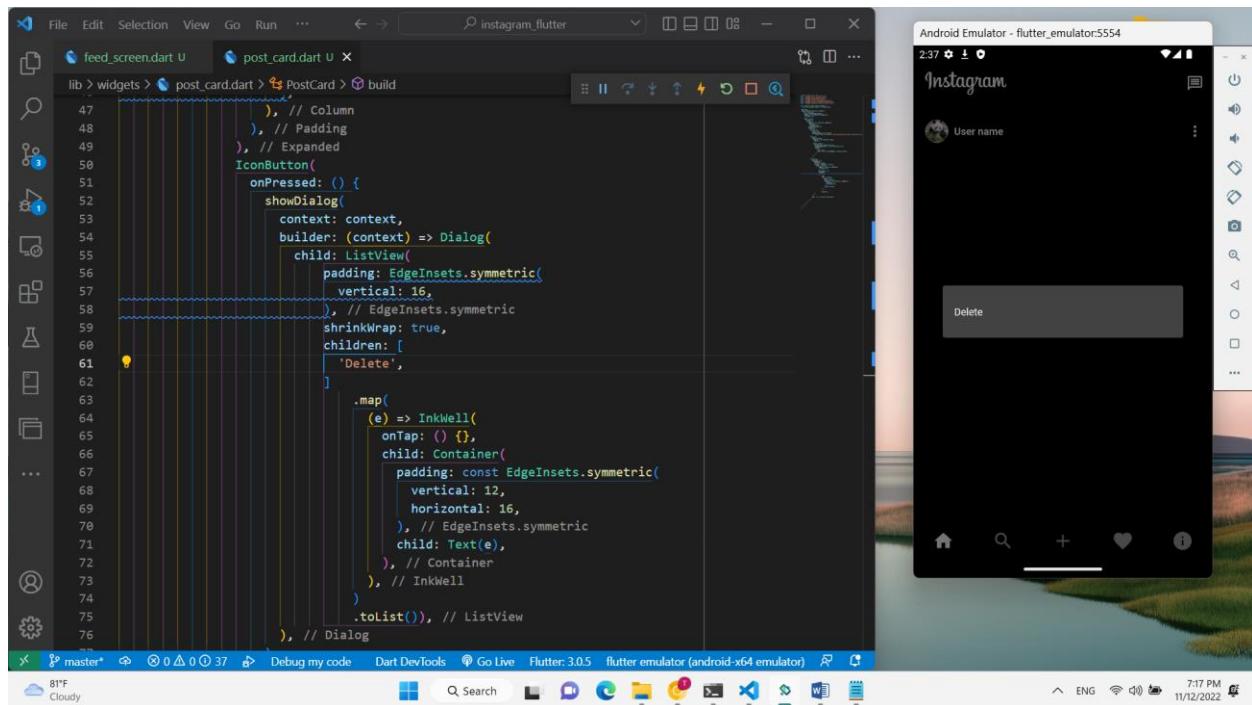
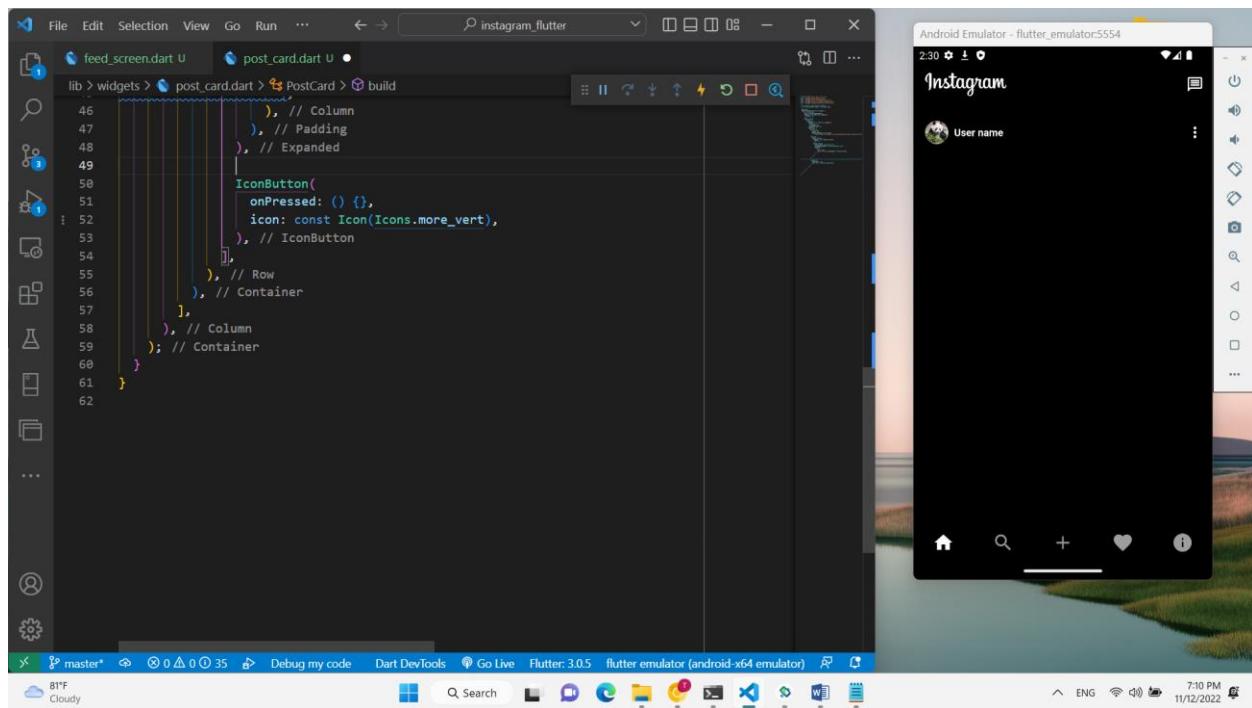


The screenshot shows a code editor interface for a Flutter application named "INSTAGRAM_FLUTTER". The current file being edited is `feed_screen.dart`. The code defines a `_FeedScreenState` class that extends `State<FeedScreen>`. It overrides the `Widget build` method to return a `Scaffold` widget. The `appBar` is set to an `AppBar` with a background color of `mobileBackgroundColor`, a title of `SvgPicture.asset('assets/ic_instagram.svg')` with a `color` of `primaryColor` and height of 32, and actions containing an `IconButton` with an `onPressed` callback and an `Icon(Icons.message_outlined)`. The `Scaffold` also contains a `body` with a `Image` widget showing a landscape image. The bottom status bar indicates the time is 6:55 PM on 11/12/2022.

```
lib > screens > feed_screen.dart > _FeedScreenState >
19
20 class _FeedScreenState extends State<FeedScreen> {
21   @override
22   Widget build(BuildContext context) {
23     return Scaffold(
24       appBar: AppBar(
25         backgroundColor: mobileBackgroundColor,
26         title: SvgPicture.asset(
27           'assets/ic_instagram.svg',
28           color: primaryColor,
29           height: 32,
30         ), // SvgPicture.asset
31         actions: [
32           IconButton(
33             onPressed: () {},
34             icon: const Icon(
35               Icons.message_outlined,
36             ), // Icon
37           ), // IconButton
38         ], // AppBar
39       ); // Scaffold
40     }
41   }
42 }
```

The screenshot shows a code editor interface for the same Flutter application. The current file being edited is `post_card.dart`. The code defines a `PostCard` class that extends `StatelessWidget`. It contains a `Container` with a `Row` child. The first child of the `Row` is a `CircleAvatar` with a radius of 16 and a `backgroundImage` of a giant panda bear. The second child is an `Expanded` widget containing a `Text` with the placeholder "User name". The bottom status bar indicates the time is 7:08 PM on 11/12/2022.

```
lib > widgets > post_card.dart > PostCard > build
21 padding: const EdgeInsets.symmetric(
22   vertical: 4,
23   horizontal: 16,
24 ).copyWith(right: 0), // EdgeInsets.symmetric
25 child: Row(
26   children: [
27     CircleAvatar(
28       radius: 16,
29       backgroundImage: NetworkImage(
30         "https://media.istockphoto.com/id/1221133425/photo/giant-panda-bear-eating-l
31       ), // CircleAvatar
32     Expanded(
33       child: Padding(
34         padding: const EdgeInsets.only(
35           left: 8,
36         ), // EdgeInsets.only
37         child: Column(
38           mainAxisAlignment: MainAxisAlignment.min,
39           crossAxisAlignment: CrossAxisAlignment.start,
40           children: [
41             Text(
42               "User name",
43               style: TextStyle(fontWeight: FontWeight.bold),
44             ), // Text
45           ],
46         ), // Column
47       ), // Padding // Expanded
48     ), // Row
49   ], // Container
50 )
```





The screenshot shows a Flutter development environment with the code for a post card screen and its resulting UI on an Android emulator.

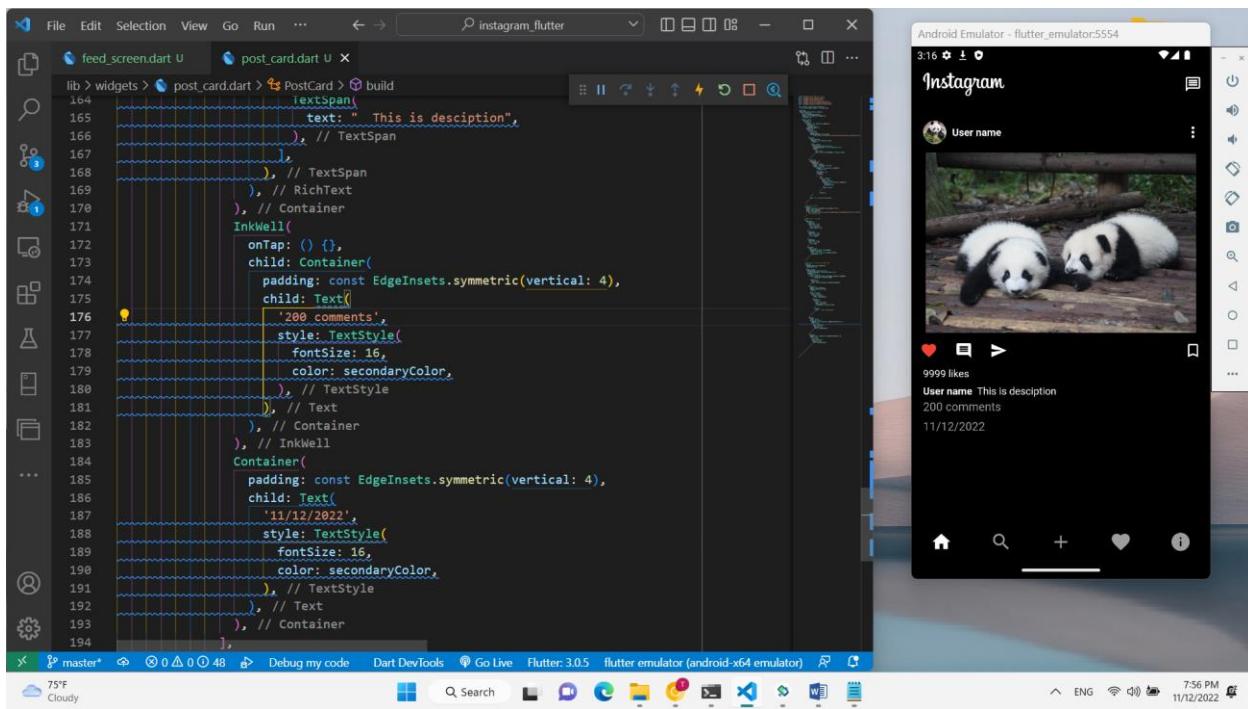
Code (feed_screen.dart):

```
lib > widgets > post_card.dart > PostCard > build
  IconButton(
    onPressed: () {},
    icon: Icon(
      Icons.favorite,
      color: Colors.red,
    ), // Icon
  ), // IconButton
  IconButton(
    onPressed: () {},
    icon: Icon(
      Icons.comment,
    ), // Icon
  ), // IconButton
  IconButton(
    onPressed: () {},
    icon: Icon(
      Icons.send,
    ), // Icon
  ), // IconButton
  Expanded(
    child: Align(
      alignment: Alignment.bottomRight,
      child: IconButton(
        icon: const Icon(Icons.bookmark_border),
        onPressed: () {},
      ), // IconButton
    ), // Align
  ), // Expanded
), // Row
```

Emulator Preview:

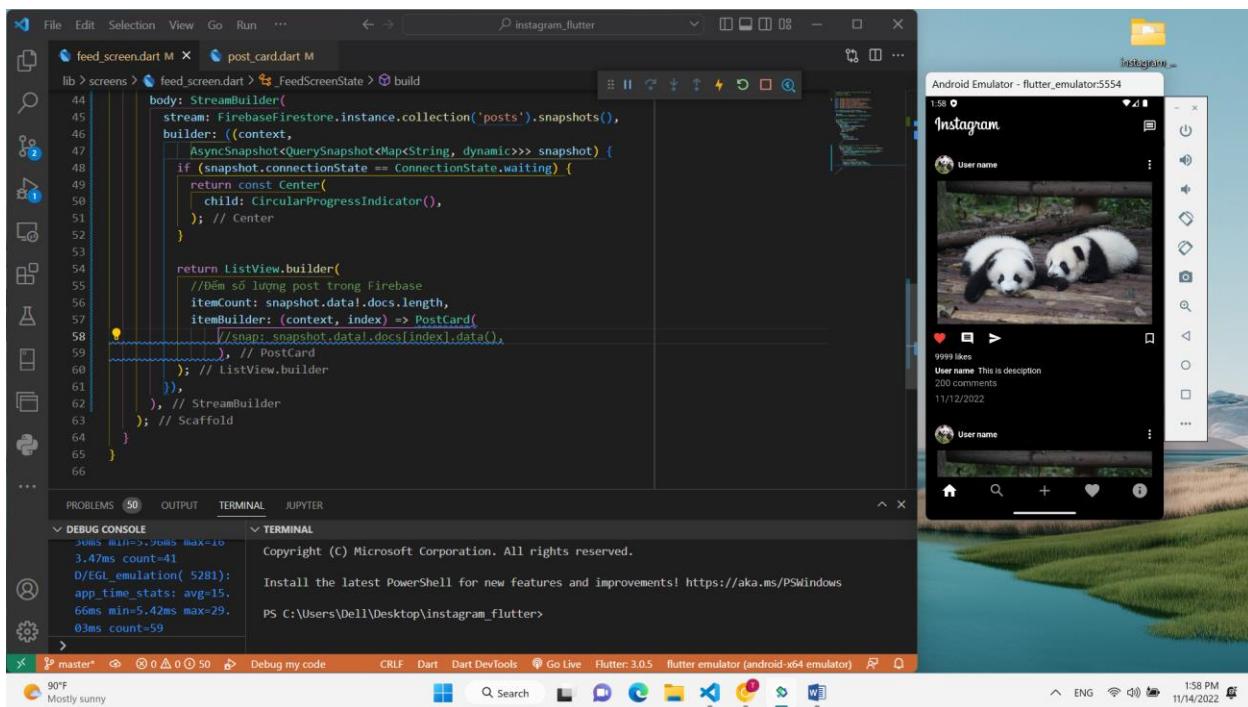
The emulator displays the Instagram app interface. A post featuring two pandas is shown, with a red heart icon indicating it has been liked. The bottom navigation bar includes icons for home, search, add, and heart.

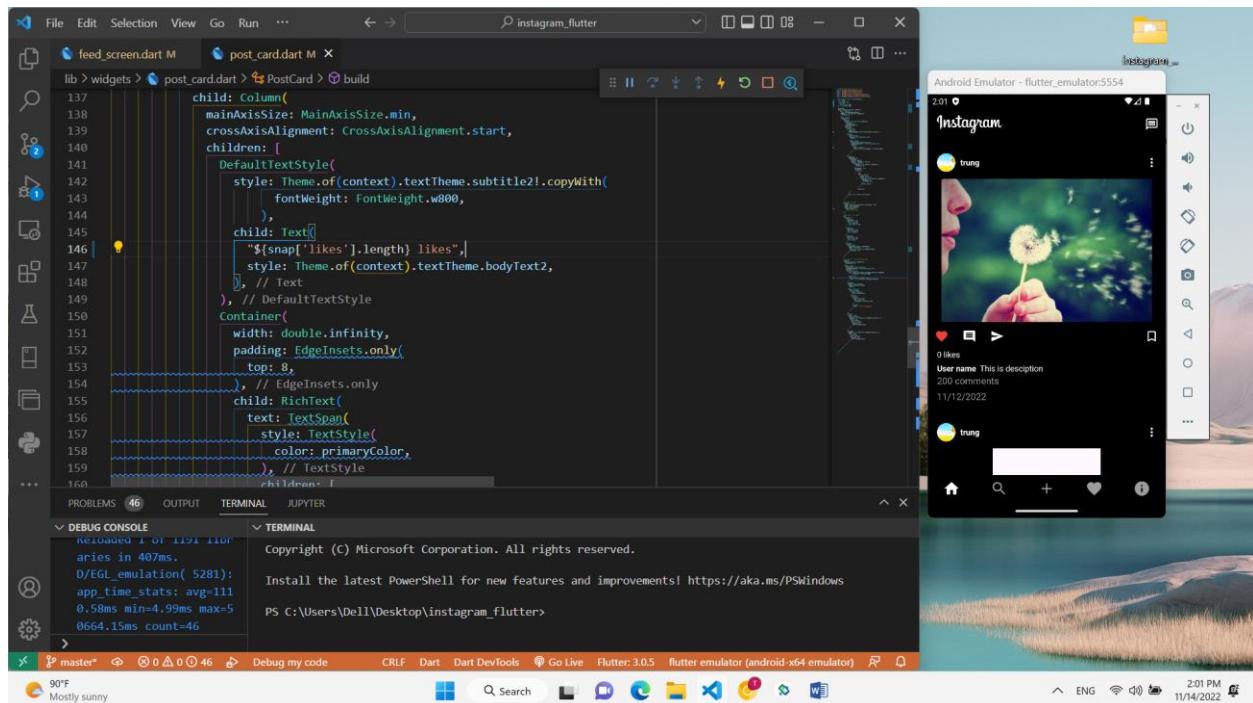
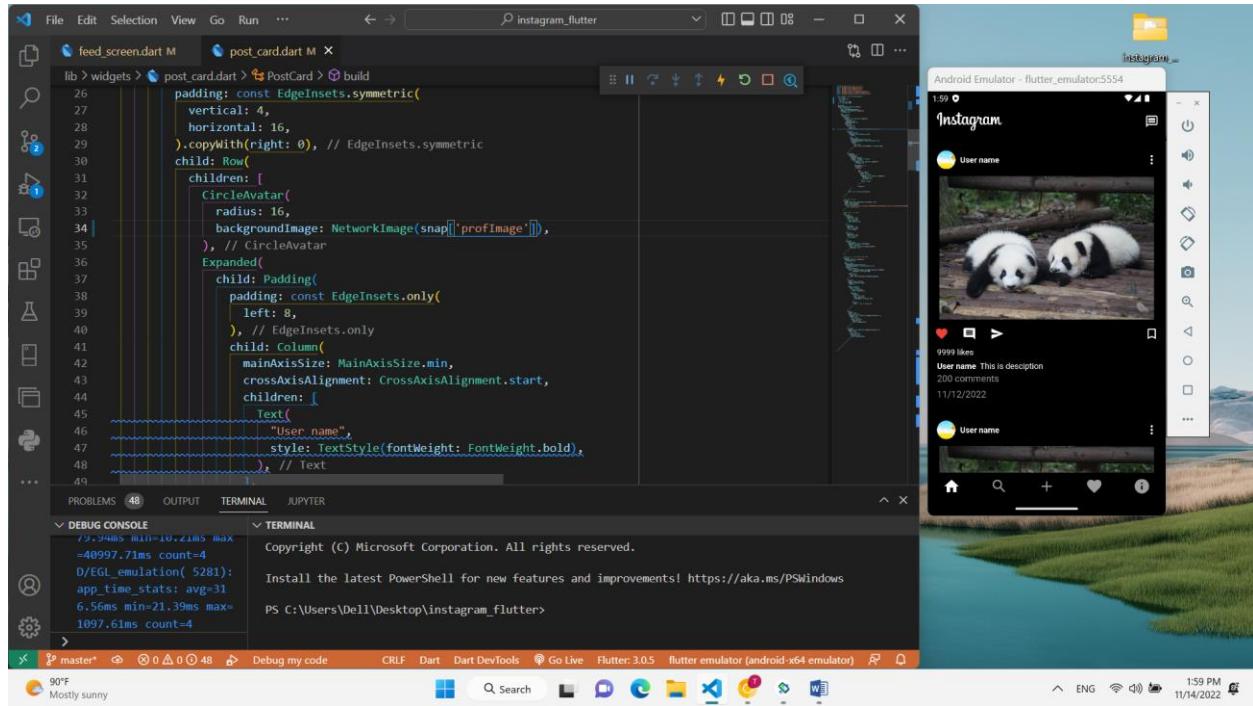
Feed Screen hoàn thành

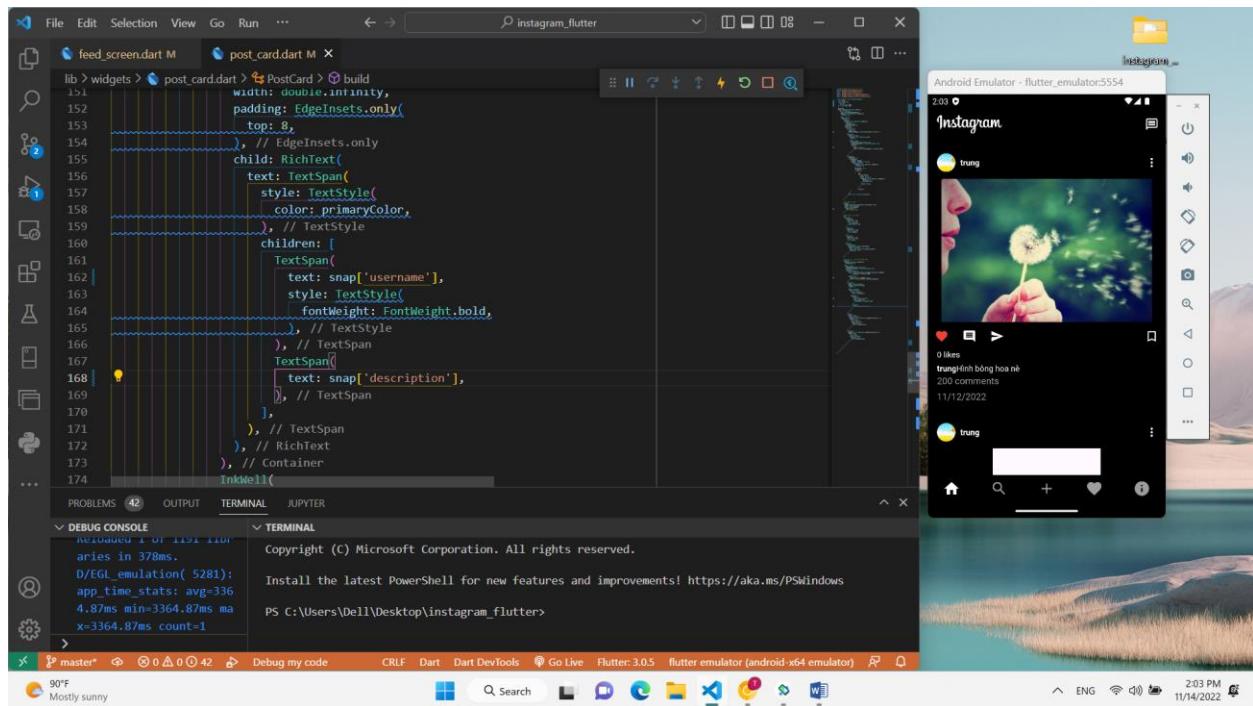
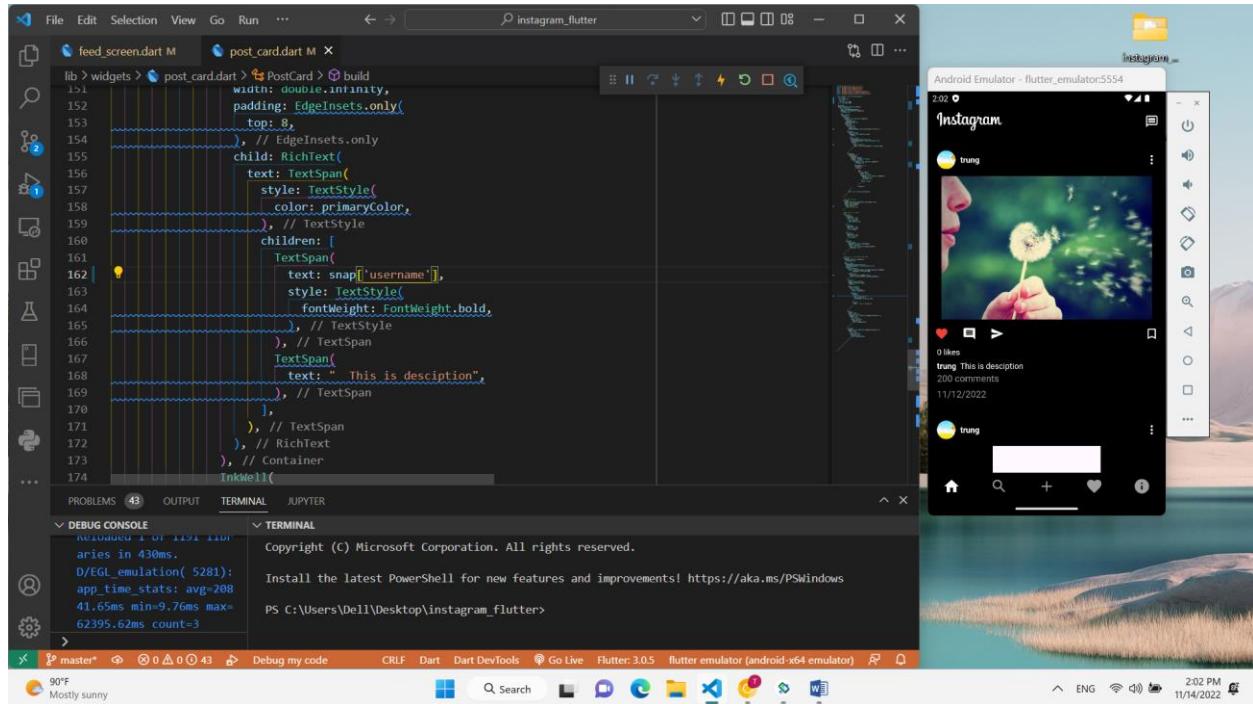


Bước 33. Lấy dữ liệu posts từ firebase

Dùng ListView

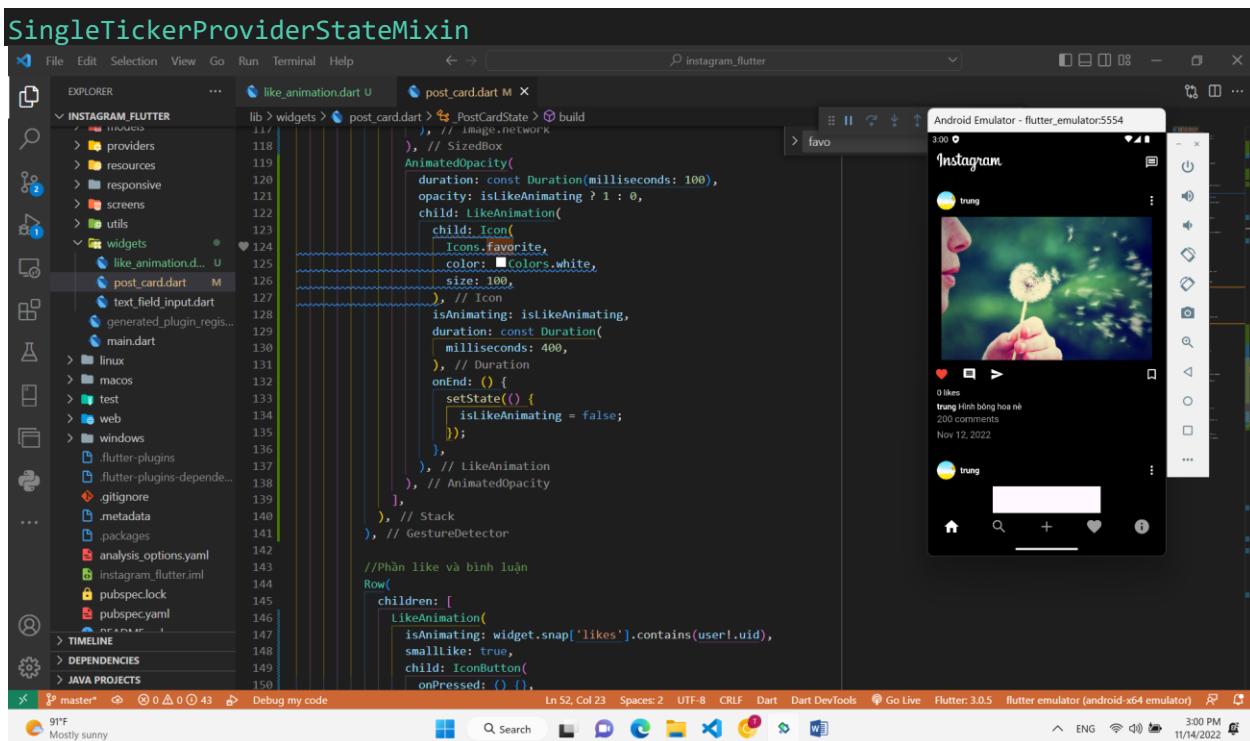




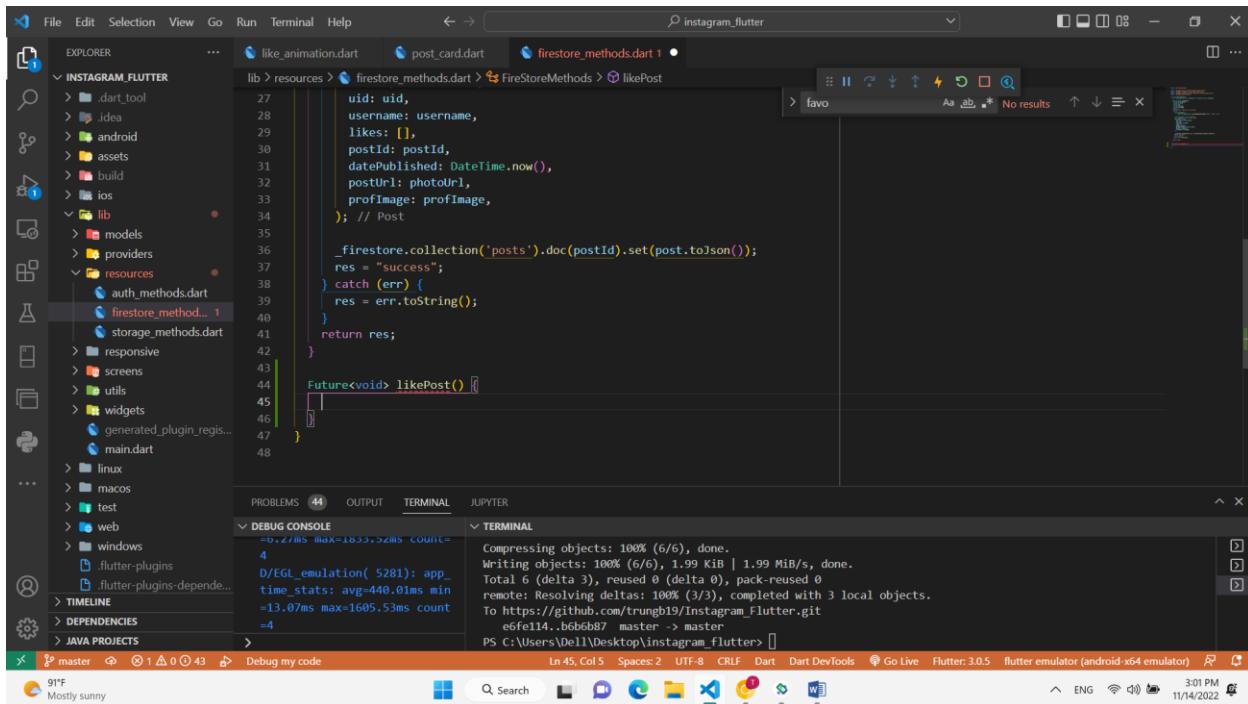


The screenshot shows a Flutter development setup in an IDE. On the left, the code editor displays `post_card.dart` with Dart code for a post card UI. The code includes various widgets like `Text`, `Image`, and `Container`. On the right, the `flutter_emulator` window shows a simulated Instagram post card. The post features a profile picture of a person blowing a dandelion, with the caption "trung". It includes metrics like 0 likes, 200 comments, and a timestamp of Nov 12, 2022. At the bottom, there's a navigation bar with icons for home, search, and other functions. The status bar at the very bottom shows weather information (90°F, mostly sunny) and system details (CRLF, Dart, Dart DevTools, Go Live, Flutter: 3.0.5, flutter emulator (android-x64 emulator)).

Bước 34. Tạo hiệu ứng like (like animation)



Bước 35. Cập nhật like post lên Firebase



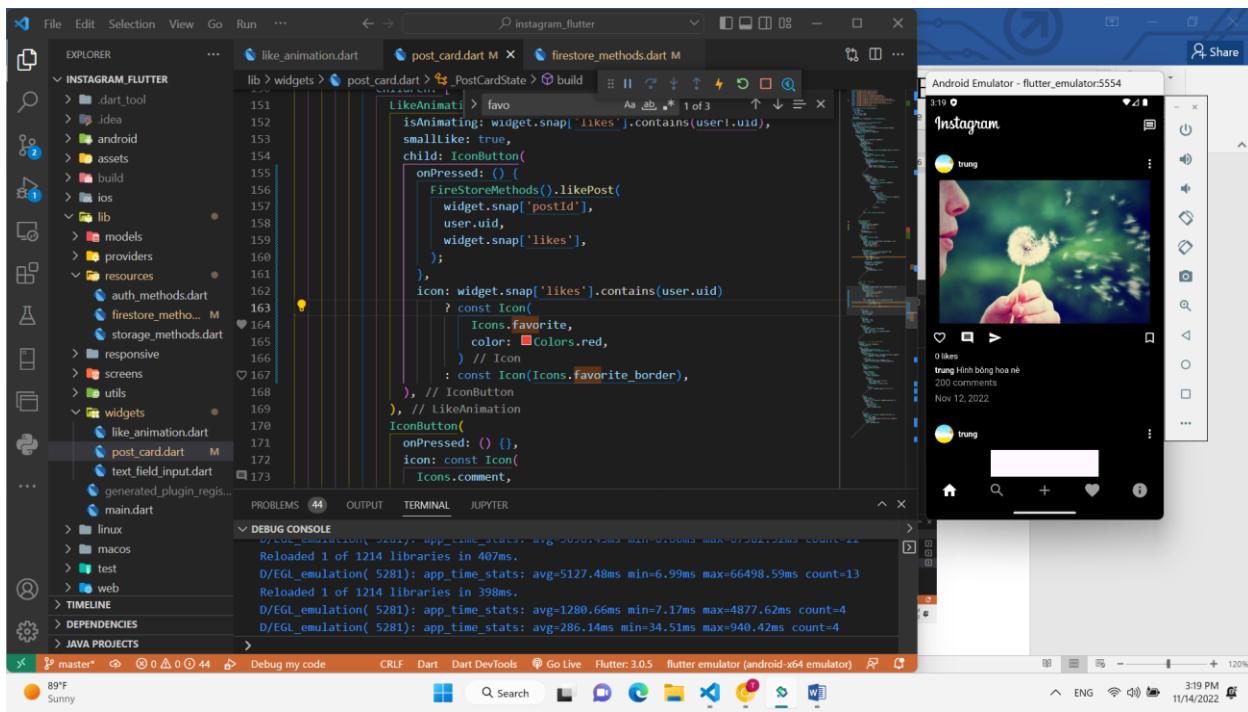
The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure for "INSTAGRAM_FLUTTER".
- Editor:** Displays the file "post_card.dart" containing Dart code for updating a post's likes in Firestore.
- Terminal:** Shows the output of the Flutter build process and the command "flutter emulator".
- Debug Console:** Shows logs related to the emulator and the current state of the application.
- Status Bar:** Shows the weather as "91°F Mostly sunny" and the date/time as "11/14/2022 3:01 PM".

```
lib > resources > firestore_methods.dart > FireStoreMethods > likePost
  uid: uid,
  username: username,
  likes: [],
  postId: postId,
  datePublished: DateTime.now(),
  postUrl: postUrl,
  profImage: profImage,
}; // Post

_firestore.collection('posts').doc(postId).set(post.toJson());
res = "success";
} catch (err) {
res = err.toString();
}
return res;
}

Future<void> likePost() {
}
```



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure for "INSTAGRAM_FLUTTER".
- Editor:** Displays the file "post_card.dart" containing Dart code for a "LikeAnimation" widget.
- Terminal:** Shows the output of the Flutter build process and the command "flutter emulator".
- Debug Console:** Shows logs related to the emulator and the current state of the application.
- Status Bar:** Shows the weather as "89°F Sunny" and the date/time as "11/14/2022 3:19 PM".

```
LikeAnimation > favo
  isAnimating: widget.snap['likes'].contains(user!.uid),
  smalllike: true,
  child: IconButton(
    onPressed: () {
      FireStoreMethods().likePost(
        widget.snap['postId'],
        user.uid,
        widget.snap['likes'],
      );
    },
    icon: widget.snap['likes'].contains(user.uid)
      ? const Icon(
          Icons.favorite,
          color: Colors.red,
        ) // Icon
      : const Icon(Icons.favorite_border),
  ), // IconButton
  IconButton(
    onPressed: () {},
    icon: const Icon(
      Icons.comment,
    ),
  ), // IconButton

```

The right side of the screen shows the Android emulator displaying an Instagram post with a dandelion image. The post has 0 likes and 200 comments. A user named "trung" is interacting with the like button.

