

# Hướng dẫn sử dụng Firebase Authentication ở Client-side bằng Python

<i>Giới thiệu về Firebase Authentication</i> .....	2
<i>Dùng Python ở client</i> .....	2
Chuẩn bị dự án Firebase .....	2
Cài đặt thư viện và khởi tạo ứng dụng .....	3
Tạo tài khoản bằng email và mật khẩu .....	4
Gửi email xác minh .....	4
Đăng nhập bằng email và mật khẩu .....	5
Xóa hoặc cập nhật tài khoản .....	5
<i>Kết hợp Authentication với các dịch vụ khác của Firebase</i> .....	5
<i>Nâng cao 1: Đăng nhập bằng Google (tài khoản Gmail)</i> .....	6
Kích hoạt nhà cung cấp Google .....	7
Thực hiện quy trình đăng nhập .....	7
<i>Nâng cao 2: Đăng nhập bằng GitHub (sinh viên tìm hiểu thêm)</i> .....	14
<i>Kết luận</i> .....	14
<i>Tham khảo</i> .....	14

## Giới thiệu về Firebase Authentication

Firebase Authentication là dịch vụ xác thực của Google cung cấp cho ứng dụng backend và SDK để sử dụng để xác định danh tính người dùng. Như chúng ta đã biết, hầu hết các ứng dụng cần biết danh tính của người dùng và việc biết được điều này giúp lưu trữ dữ liệu an toàn trên đám mây và đồng bộ trải nghiệm trên nhiều thiết bị [1]. Firebase Authentication cung cấp các SDK và thư viện giao diện người dùng có sẵn để xác thực người dùng với nhiều phương thức, bao gồm:

- Email và mật khẩu: SDK cung cấp các phương thức tạo và quản lý tài khoản sử dụng email/mật khẩu và tự động gửi email đặt lại mật khẩu.
- Số điện thoại: gửi mã xác minh qua SMS.
- Các nhà cung cấp bên thứ ba liên kết: Google, Facebook, Twitter, GitHub... được tích hợp sẵn.
- Đăng nhập ẩn danh, cho phép tạo tài khoản tạm thời và nâng cấp thành tài khoản thực sau này.

Firebase Authentication tích hợp chặt chẽ với các dịch vụ khác của Firebase như Realtime Database, Storage, FireStore [1]...

## Dùng Python ở client

Firebase cung cấp SDK chính thức cho web (JavaScript), Android, iOS, Flutter, C++, Unity... nhưng không hỗ trợ trực tiếp Python trên client. Tuy nhiên, đối với các ứng dụng Python chạy trên máy người dùng (ví dụ: ứng dụng desktop, script) vẫn có thể sử dụng **Firebase Authentication REST API**. Để sử dụng Firebase REST API này, chúng ta sử dụng **requests library** để gọi các API endpoints của Firebase. Thư viện này là một tập lệnh đơn giản, tạo sẵn và hỗ trợ chúng ta thực hiện các thao tác tạo tài khoản, đăng nhập, gửi email xác minh... thông qua phương thức HTTP.

**Lưu ý an toàn:** API Key trong cấu hình ứng dụng không phải bí mật tuyệt đối nhưng nên thiết lập [kiểm soát miền](#) để hạn chế lạm dụng.

### Chuẩn bị dự án Firebase

- **Tạo/đăng nhập dự án trên Firebase Console:** vào <https://console.firebaseio.google.com>, tạo một dự án mới hoặc chọn dự án hiện có.
- **Đăng ký ứng dụng Web:** trong phần **Project settings** → **General**, nhấn “Add app” và chọn “Web app”. Đặt tên, sau đó Firebase sẽ cung cấp mã cấu hình gồm apiKey, authDomain, databaseURL, projectId, storageBucket, messagingSenderId, appId... Lưu tạm cấu hình này, chúng ta có thể dùng cho Python ở những bước sau. Ví dụ:

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
```

```

import { getAnalytics } from "firebase/analytics";
const firebaseConfig = {
  apiKey: "AIzaSyBjwvTagbmM0E1",
  authDomain: "module11-firebase.firebaseio.com",
  databaseURL: "https://module11-firebase.firebaseio.com",
  projectId: "module11-firebase",
  storageBucket: "module11-firebase.appspot.com",
  messagingSenderId: "964743",
  appId: "1:ea91684aa",
  measurementId: "G-LS3YF"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);

```

- **Kích hoạt phương thức đăng nhập:** vào mục **Build → Authentication → Sign-in method.** Bật “Email/Password” hoặc các nhà cung cấp khác tùy nhu cầu.
- **Tạo tài khoản dịch vụ (nếu cần gọi các API đặc quyền):** vào mục **Project settings → Service accounts** và tạo khóa JSON. Khóa này dùng cho Admin SDK để xác minh token hoặc quản lý người dùng trên server, không dùng ở client.

## Cài đặt thư viện và khởi tạo ứng dụng

Cài đặt thư viện firebase-rest-api bằng pip:

```
pip install firebase-rest-api
```

Tạo tệp Python, nhập thư viện và khởi tạo ứng dụng bằng cấu hình Firebase. Cấu hình config được lấy ở bước đăng ký Web app:

```

import firebase
import json

# Cấu hình Firebase (thay bằng giá trị thật của các bạn)
config = {
  "apiKey": "your API key",
  "authDomain": "module11-firebase.firebaseio.com",
  "databaseURL": "https://module11-firebase-default-rtdb.firebaseio.com",
  "projectId": "module11-firebase",
  "storageBucket": "module11-firebase.appspot.com",
  "messagingSenderId": "123456789",

```

```
        "appId": "1:123456789:web:abcdef123456"  
    }  
  
    # Khởi tạo ứng dụng Firebase  
    app = firebase.initializeApp(config)  
  
    # Khởi tạo dịch vụ Authentication  
    auth = app.auth()
```

Thư viện cho phép truy cập Realtime Database, Firestore, Storage... tuy nhiên hướng dẫn này tập trung vào Authentication.

## Tạo tài khoản bằng email và mật khẩu

Firebase Authentication hỗ trợ tạo người dùng email/mật khẩu và quản lý mật khẩu [2].

```
def test_sign_up():  
    print("==== TEST SIGN UP ===")  
    email = "test2@example.com"  
    password = "password123"  
    try:  
        result = auth.create_user_with_email_and_password(email, password)  
        print(f"Đăng ký thành công!")  
        print(f"Kết quả: {json.dumps(result, indent=2, ensure_ascii=False)}")  
        return result  
    except Exception as e:  
        print(f"Đăng ký thất bại: {e}")  
        return None
```

Kết quả trả về là dictionary chứa email, idToken, refreshToken, localId... idToken (JWT) dùng để xác thực mỗi khi gọi API Firebase khác.

## Gửi email xác minh

Sau khi tạo người dùng, nên yêu cầu xác minh địa chỉ email. Dùng `send_email_verification(id_token)` để Firebase gửi email xác minh tới người dùng [2]:

```
id_token = result['idToken']  
auth.send_email_verification(id_token)
```

Người dùng cần nhấp vào liên kết trong email để xác minh. Bạn có thể kiểm tra trạng thái xác minh bằng cách gọi `get_account_info(id_token)`[\[2\]](#).

## Đăng nhập bằng email và mật khẩu

Để người dùng đăng nhập, sử dụng `sign_in_with_email_and_password(email, password)`[\[2\]](#). Phương thức này xác thực thông tin và trả về token xác thực:

```
def test_sign_in():
    print("\n==== TEST SIGN IN ====")
    email = "test2@example.com"
    password = "password123"

    try:
        result = auth.sign_in_with_email_and_password(email, password)
        print(f"Đăng nhập thành công!")
        print(f"Kết quả: {json.dumps(result, indent=2, ensure_ascii=False)}")
        return result
    except Exception as e:
        print(f"Đăng nhập thất bại: {e}")
    return None
```

Kết quả là dictionary chứa `idToken`, `refreshToken`, `expiresIn...` Token này nên lưu tạm và sử dụng khi truy cập các dịch vụ Firebase khác (ví dụ truy vấn Realtime Database hoặc Storage). Khi token hết hạn, dùng `auth.refresh(refresh_token)`[\[2\]](#) để lấy token mới:

```
refreshed = auth.refresh(user['refreshToken'])
new_id_token = refreshed['idToken']
```

## Xóa hoặc cập nhật tài khoản

- **Xóa người dùng:** dùng `delete_user_account(id_token)`[\[2\]](#) để xóa tài khoản tương ứng.
- **Thay đổi mật khẩu:** dùng `change_password(id_token, new_password)`[\[2\]](#).
- **Thay đổi email:** dùng `change_email(id_token, new_email)`[\[2\]](#).
- **Cập nhật tên hiển thị hoặc ảnh đại diện:** dùng `update_profile(id_token, display_name, photo_url)`[\[2\]](#).
- **Đăng nhập ẩn danh:** dùng `sign_in_anonymous()` để tạo tài khoản tạm thời[\[2\]](#).

## Kết hợp Authentication với các dịch vụ khác của Firebase

Sau khi xác thực, bạn có thể dùng `idToken` để làm bằng chứng truy cập dữ liệu hoặc gọi API khác. Ví dụ, lưu thông tin người dùng vào Realtime Database:

```

def test_database_operations(user_data):
    if not user_data:
        print("Không có user data để test database!")
        return

    print("\n==== TEST FIREBASE DATABASE ====")

    # Khởi tạo database
    db = app.database()

    # Dữ liệu để lưu
    data = {
        "name": "Test User",
        "email": user_data.get('email'),
        "timestamp": "2025-01-21"
    }

    try:
        # Lưu dữ liệu vào Firebase Realtime DB: db > users > new_user
        result = db.child("users").push(data, user_data.get('idToken'))
        print(f"Lưu dữ liệu thành công!")
        print(f"Key: {result}")

        # Đọc dữ liệu từ database
        users = db.child("users").get(user_data.get('idToken'))
        print(f"Đọc dữ liệu thành công!")
        print(f"Dữ liệu: {users.val()}")

    except Exception as e:
        print(f"Database thất bại: {e}")

```

\***Sinh viên tìm hiểu thêm về Firebase Storage.**

## Nâng cao 1: Đăng nhập bằng Google (tài khoản Gmail)

Tương tự GitHub, Firebase hỗ trợ đăng nhập với tài khoản Google. Việc sử dụng Google làm nhà cung cấp liên kết cho phép người dùng đăng nhập nhanh bằng Gmail và lấy thông tin hồ sơ. Trong danh sách các nhà cung cấp, Google nằm cùng nhóm với Facebook, Twitter và GitHub [3].

## Kích hoạt nhà cung cấp Google

- Vào **Firebase Console** → **Authentication** → **Sign-in method** và chọn **Google**. Trong mục “Sign-in providers”, bật Google và nhấn **Save**.
- Nếu cần quyền truy cập API nâng cao, bạn có thể cấu hình OAuth 2.0 client trên Google Cloud Console; tuy nhiên, với Firebase Authentication, việc bật provider Google trên console là đủ và bạn không cần Client Secret như GitHub [3].

## Thực hiện quy trình đăng nhập

`firebase-rest-api` cung cấp sẵn phương thức `authenticate_login_with_google()` để lấy URL đăng nhập với Google [2]. Sau khi người dùng đăng nhập, bạn vẫn sử dụng `sign_in_with_oauth_credential` để đổi callback URL thành token [2].

Đầu tiên bạn cần thêm “*Web client ID*” và “*Web client secret*” vào cấu hình mã nguồn. Thực hiện các bước sau để lấy 2 thông tin này:

- Vào Google Cloud Console: <https://console.cloud.google.com/>
- Chọn project của chúng ta, ví dụ module11-firebase (hoặc tạo project mới)
- Kích hoạt Google+ API
  - o Vào APIs & Services → Library
  - o Tìm kiếm "Google+ API" hoặc "Google Identity"
  - o Click Enable
- Tạo OAuth 2.0 Credentials
  - o Vào APIs & Services → Credentials
  - o Click + CREATE CREDENTIALS → OAuth client ID
  - o Chọn Web application
  - o Điền thông tin:
    - Name: Firebase Web App
    - Authorized JavaScript origins:
      - <http://localhost:3000>
      - <https://module11-firebase.firebaseio.com>
    - Authorized redirect URLs: <https://module11-firebase.firebaseio.com>
- Lấy Credentials: Sau khi tạo xong, bạn sẽ thấy:
  - o Client ID: 923865474355-xxxxxx.apps.googleusercontent.com
  - o Client Secret: GOCSXP-xxxxxxxxxxxxxxxxxxxxxx
- Cập nhật Firebase Console
  - o Vào Firebase Console → Authentication → Sign-in method
  - o Click Google → Enable
  - o Paste Client ID và Client Secret vào
  - o Click Save

Chúng ta tiến hành khai báo thư viện, cấu hình tham số, và tạo webserver để xử lý hàm callback sau khi đăng nhập xong từ Google:

```
import firebase
import json

import webbrowser
import threading
import time
from http.server import HTTPServer, BaseHTTPRequestHandler
from urllib.parse import urlparse, parse_qs

# Cấu hình Firebase
config = {
    "apiKey": "AIzaSyBYmk_gbmM0E1nI",
    "authDomain": "module11-firebase.firebaseio.com",
    "databaseURL": "https://module11-firebase.firebaseio.com",
    "projectId": "module11-firebase",
    "storageBucket": "module11-firebase.appspot.com",
    "messagingSenderId": "92384355",
    "appId": "1:92374355:web:c550d684aa4a46"
}

# Cấu hình Google OAuth Client Secret
google_client_secret = {
    "client_id": "923rif6p.apps.googleusercontent.com",
    "client_secret": "zpaMCY_JdRM0U",
    "redirect_uris": [
        "http://localhost:8000/callback"
    ]
}

# Khởi tạo ứng dụng Firebase
app = firebase.initialize_app(config)
auth = app.auth(client_secret=google_client_secret)

# Global variables for callback handling
callback_received = False
callback_url = None
callback_data = None

class CallbackHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        global callback_received, callback_url, callback_data
```

```
# Parse URL
parsed_url = urlparse(self.path)
query_params = parse_qs(parsed_url.query)

# Check if this is a Google OAuth callback
if 'code' in query_params and 'state' in query_params:
    callback_received = True
    callback_url = f"http://localhost:8000{self.path}"
    callback_data = {
        'code': query_params['code'][0],
        'state': query_params['state'][0]
    }

# Send response to browser
self.send_response(200)
self.send_header('Content-type', 'text/html')
self.end_headers()

html_response = """
<!DOCTYPE html>
<html>
<head>
    <title>Google Sign-in Success</title>
    <meta charset="UTF-8">
    <style>
        body { font-family: Arial, sans-serif; text-align: center;
padding: 50px; }
        .success { color: #4CAF50; font-size: 24px; }
        .message { color: #666; margin-top: 20px; }
    </style>
</head>
<body>
    <div class="success">Đăng nhập Google thành công ✓ </div>
    <div class="message">Bạn có thể đóng tab này và quay lại
terminal.</div>
<script>
    setTimeout(function() {
        window.close();
    }, 3000);
</script>
```

```
</body>
</html>
"""

self.wfile.write(html_response.encode('utf-8'))

print(f"Đã nhận callback URL: {callback_url}")
print("Đang xử lý đăng nhập...")

else:
    # Not a callback, show welcome page
    self.send_response(200)
    self.send_header('Content-type', 'text/html')
    self.end_headers()

    html_response = """
<!DOCTYPE html>
<html>
<head>
    <title>Firebase Auth Callback Server</title>
    <meta charset="UTF-8">
    <style>
        body { font-family: Arial, sans-serif; text-align: center;
padding: 50px; }
        .info { color: #2196F3; font-size: 18px; }
    </style>
</head>
<body>
    <div class="info">⌚ Đang chờ callback từ Google...</div>
</body>
</html>
"""

    self.wfile.write(html_response.encode('utf-8'))

def log_message(self, format, *args):
    # Suppress default logging
    pass

def start_callback_server():
    """Khởi động web server để xử lý callback"""

```

```

global callback_received, callback_url, callback_data

server = HTTPServer(('localhost', 8000), CallbackHandler)
print("🌐 Khởi động callback server tại http://localhost:8000")

# Start server in a separate thread
server_thread = threading.Thread(target=server.serve_forever)
server_thread.daemon = True
server_thread.start()

return server

```

Sử dụng các hàm để mở ứng dụng trình duyệt, đăng nhập và chứng thực

Sau khi chứng thực thì mở và đọc Realtime Database:

```

def google_signin():
    print("\n==== GOOGLE SIGN IN ====")

    try:
        # Khởi động callback server
        server = start_callback_server()

        # Tạo URL đăng nhập Google
        login_url = auth.authenticate_login_with_google()
        print("Mở trình duyệt và đăng nhập Google tại:", login_url)
        webbrowser.open(login_url)

        # Chờ callback từ web server
        print("Đang chờ callback từ Google...")
        timeout = 60 # 60 seconds timeout
        start_time = time.time()

        while not callback_received and (time.time() - start_time) < timeout:
            time.sleep(0.1)

        if not callback_received:
            print("Timeout! Không nhận được callback từ Google")
            server.shutdown()
    
```

```
        return None

    # Đăng nhập với callback URL
    result = auth.sign_in_with_oauthCredential(callback_url)
    print(f"Đăng nhập Google thành công!")
    print(f"User: {result.get('email')}")

    # Tự động xử lý callback sau khi đăng nhập thành công
    print("\nTự động xử lý callback...")
    handle_google_signin_callback(result)

    # Shutdown server
    server.shutdown()

    return result

except Exception as e:
    print(f"Đăng nhập Google thất bại: {e}")
    return None

def handle_google_signin_callback(user_data):
    """Xử lý callback sau khi đăng nhập Google thành công"""
    print("\n==== CALLBACK HANDLER ====")

    # Test database operations
    test_database_operations(user_data)

    # Lưu user data (có thể tắt không dùng)
    save_to_file = True # Đặt False để không lưu file
    if save_to_file:
        with open("google_user_data.json", "w") as f:
            json.dump(result, f, indent=2, ensure_ascii=False)
        print("Đã lưu user data vào file: google_user_data.json")
    else:
        print("Không lưu user data vào file")

def test_database_operations(user_data):
    if not user_data:
        print("Không có user data để test database!")
```

```
    return

print("\n==== TEST FIREBASE DATABASE ===")

# Khởi tạo database
db = app.database()

try:
    # Đọc dữ liệu từ database
    users = db.child("users").get(user_data.get('idToken'))
    print(f"Đọc dữ liệu thành công!")
    print(f"Dữ liệu: {users.val()}")

except Exception as e:
    print(f"Database thất bại: {e}")

def load_google_user_data():
    """Load user data từ file đã lưu"""
    try:
        with open("google_user_data.json", "r") as f:
            user_data = json.load(f)
        print("Đã load user data từ file: google_user_data.json")
        return user_data
    except FileNotFoundError:
        print("Không tìm thấy file google_user_data.json")
        return None
    except Exception as e:
        print(f"Lỗi khi load user data: {e}")
        return None

def main():
    print("Firebase REST API Package – Complete Solution")

    # Tùy chọn: có load file JSON không? (nếu có thì bỏ dòng dưới)
    # google_user = load_google_user_data()

    google_user = None
    if not google_user:
        print("Đăng nhập Google mới...")
        google_user = google_signin()
```

```
if __name__ == "__main__":
    main()
```

### Một số giải thích:

1. *authenticate\_login\_with\_google()* gọi tới máy chủ OAuth của Google và trả về đường dẫn đăng nhập [2]. Bạn có thể mở đường dẫn này trong trình duyệt để người dùng xác thực.
2. Sau khi người dùng cấp quyền, Google sẽ chuyển hướng tới URL callback (theo định dạng `https://<projectId>.firebaseapp.com/__auth/handler?...`). Sao chép toàn bộ đường dẫn và truyền vào *sign\_in\_with\_oauth\_credential* để đổi lấy token [2].
3. Bạn có thể lưu *idToken* và *refreshToken* để truy cập các dịch vụ Firebase khác hoặc gửi lên backend (sinh viên có thể tìm hiểu thêm về lưu trữ dữ liệu backend). Xử lý callback được xử lý bởi server ở local với việc thêm `localhost` vào danh sách *Authorized domains* trong Firebase Console.

## Nâng cao 2: Đăng nhập bằng GitHub (sinh viên tìm hiểu thêm)

### Kết luận

Firebase Authentication cung cấp nền tảng xác thực linh hoạt với nhiều phương thức, dễ dàng tích hợp với các dịch vụ Firebase khác. Dù Firebase không có SDK chính thức cho Python trên client, bạn vẫn có thể sử dụng các REST API hoặc thư viện wrapper như `firebase-rest-api` để tạo, đăng nhập và quản lý người dùng trong ứng dụng Python.

### Tham khảo

[1] Firebase Authentication, <https://firebase.google.com/docs/auth>

[2] Firebase REST API 1.11.0:

<https://firebase-rest-api.readthedocs.io/en/latest/guide/authentication.html>

[3] Federated identity provider integration:

<https://firebase.google.com/docs/auth#:~:text=Federated%20identity%20provider%20integration>