EXPERIMENT NO: - 06

Name:- Laksh Sodhai **Class:-** D15A **Roll:No: -** 57

AIM: - To connect Flutter UI with Firebase database.

Theory: -

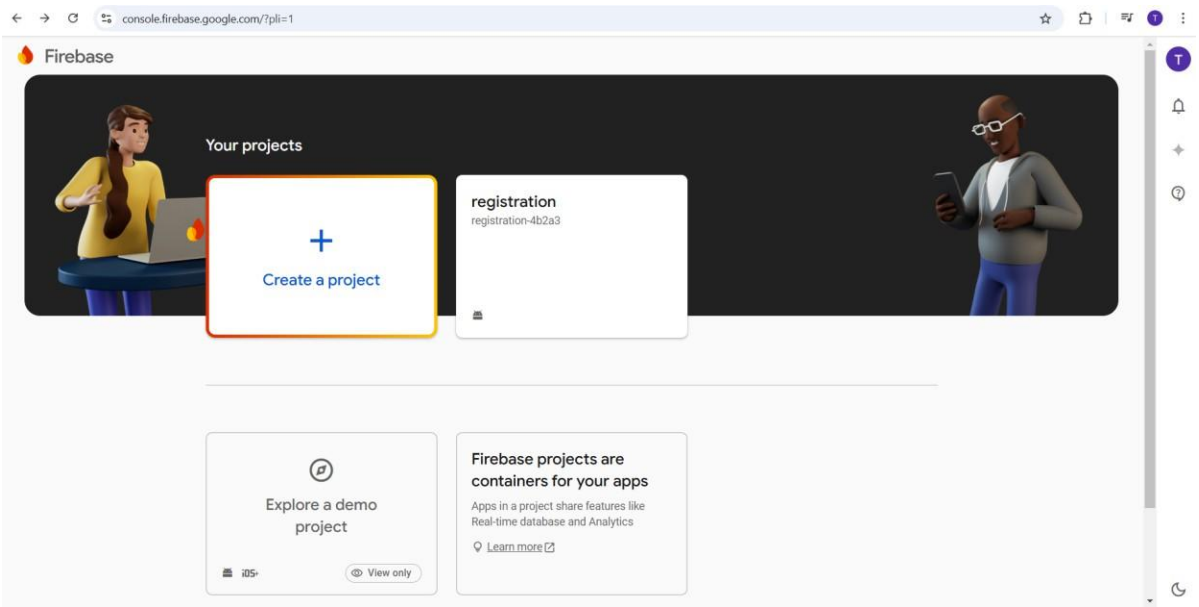
Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Firebase, a Backend-as-a-Service (BaaS) platform, provides real-time database, authentication, and cloud storage services, making it a powerful backend solution for Flutter applications.

By integrating Firebase with Flutter, developers can store and retrieve data in real time, authenticate users, and manage cloud-based data efficiently. This is particularly useful for applications requiring dynamic content updates and user interactions.

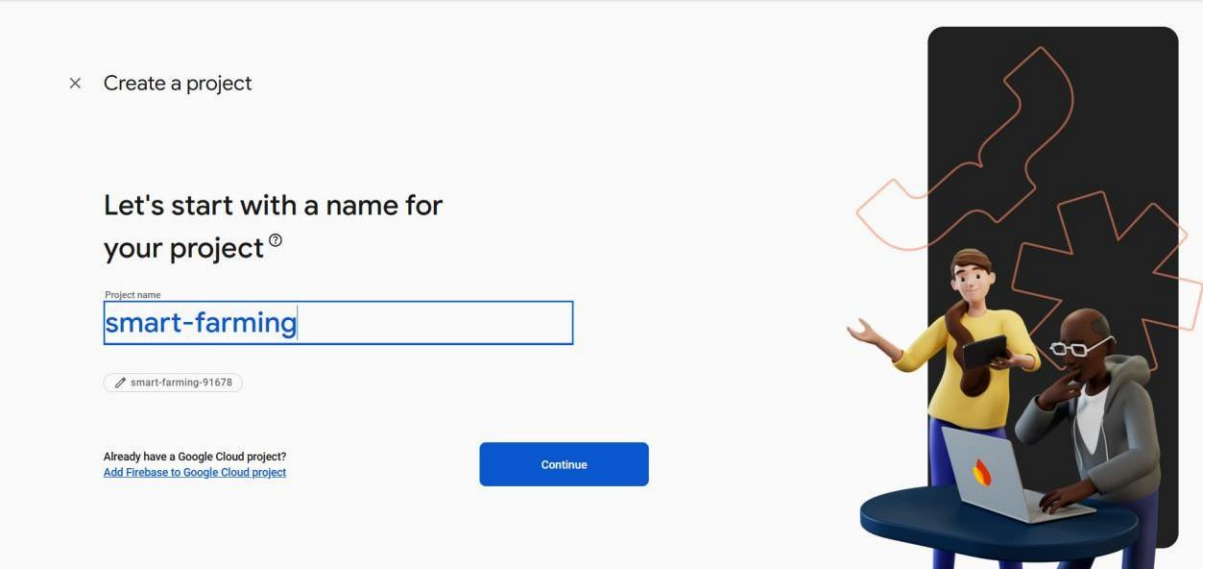
- **Steps to Connect Flutter UI with Firebase Database**

Step 1:

1.1) Go to Firebase Console and Create a Firebase Project

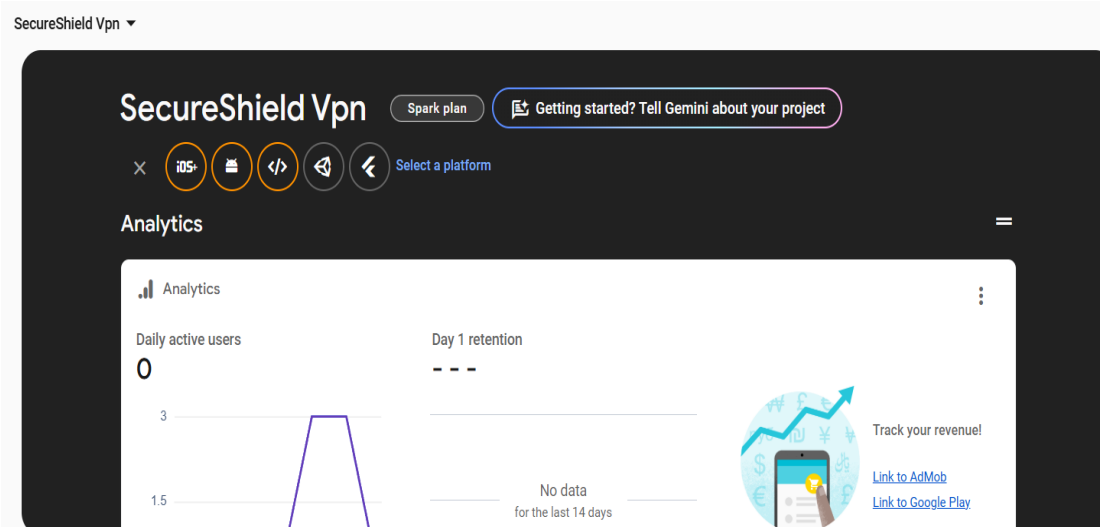


1.2) Click on Create a Project and give it a suitable name.

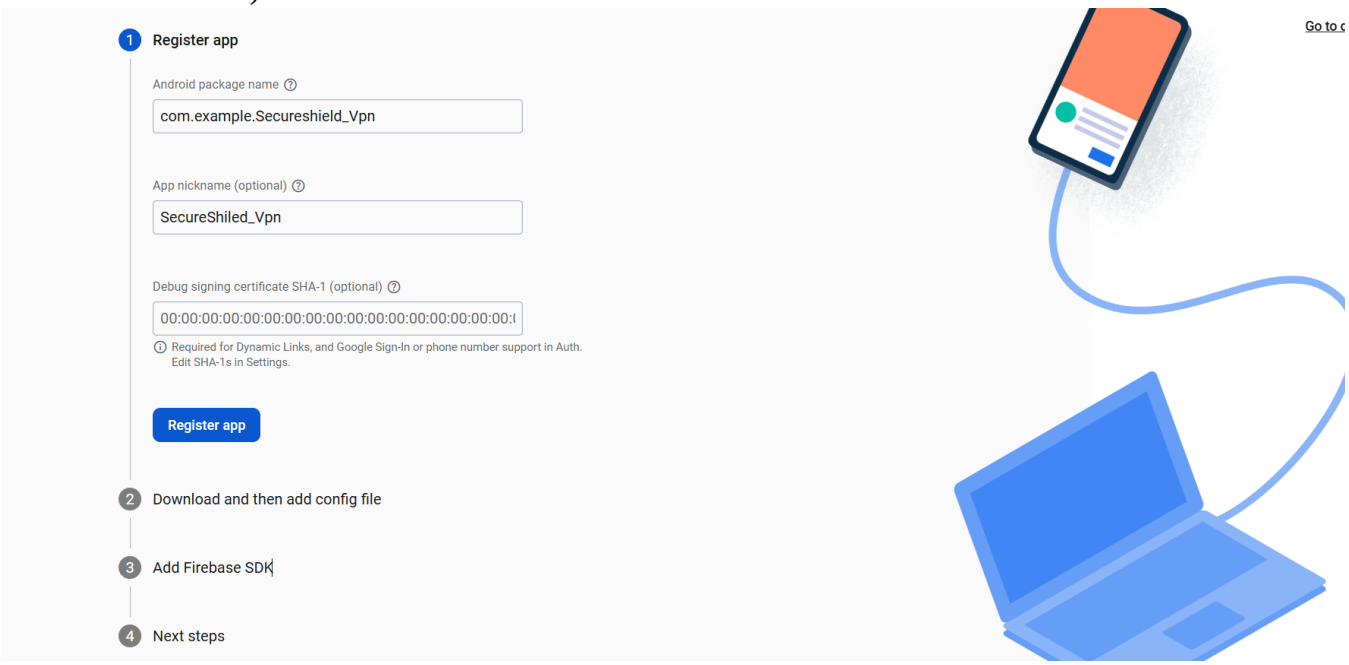


Step 2:- Add Firebase to Your Flutter App

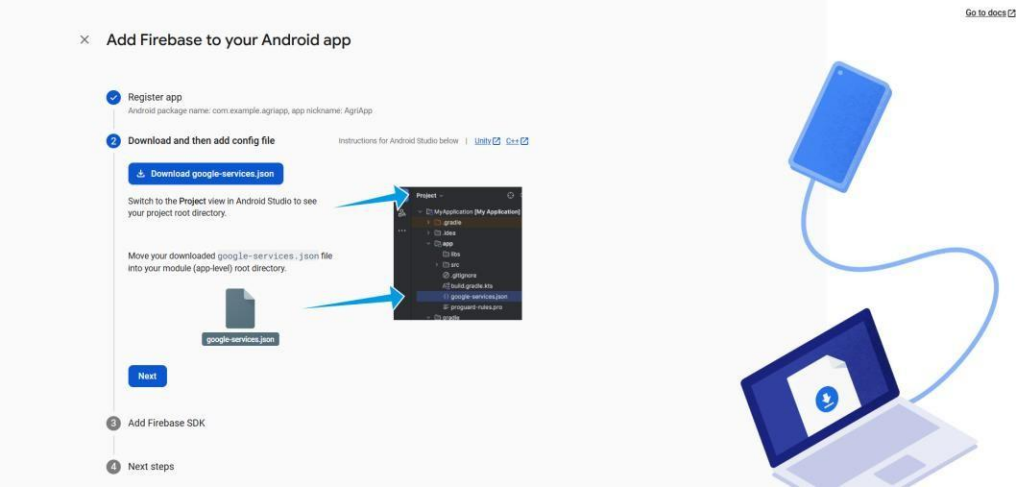
2.1) Click on Android/iOS/Web based on your Flutter application

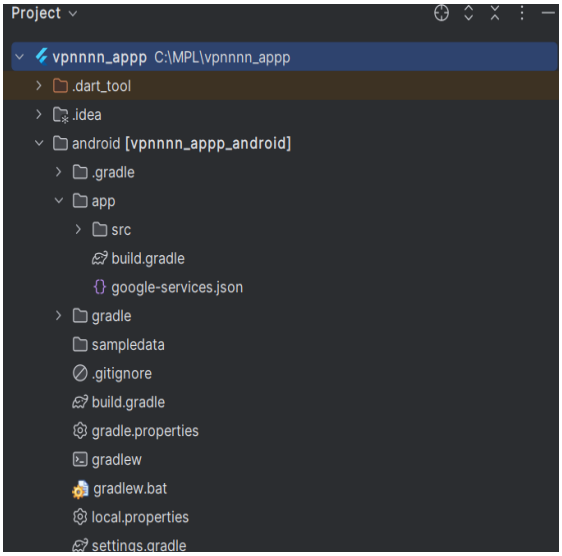


2.2) Register your app with a unique package name (found in android/app/build.gradle for Android).

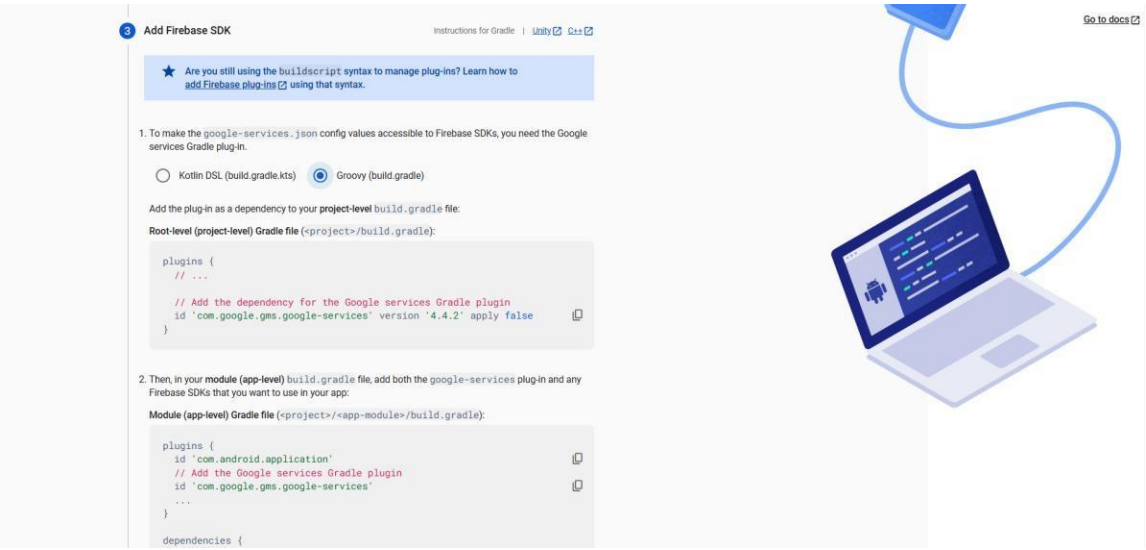


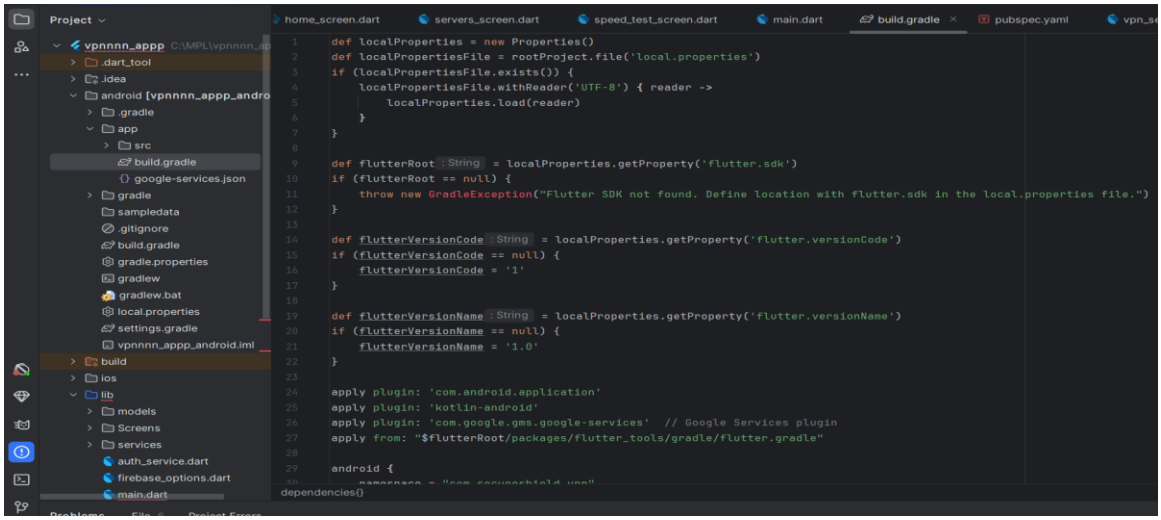
2.3) Download the google-services.json (for Android) & place the JSON file inside android/app/ directory.





2.4) Add Firebase SDK dependencies to android/build.gradle



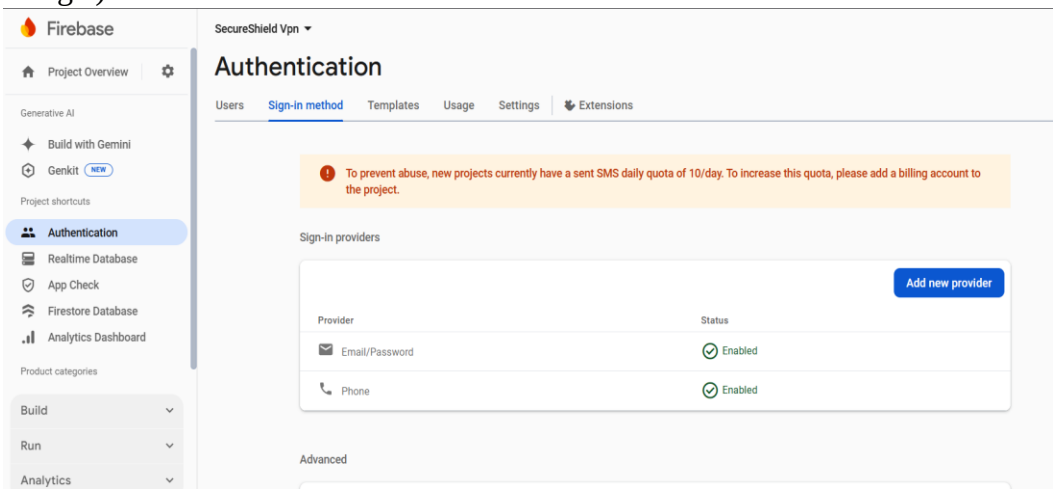


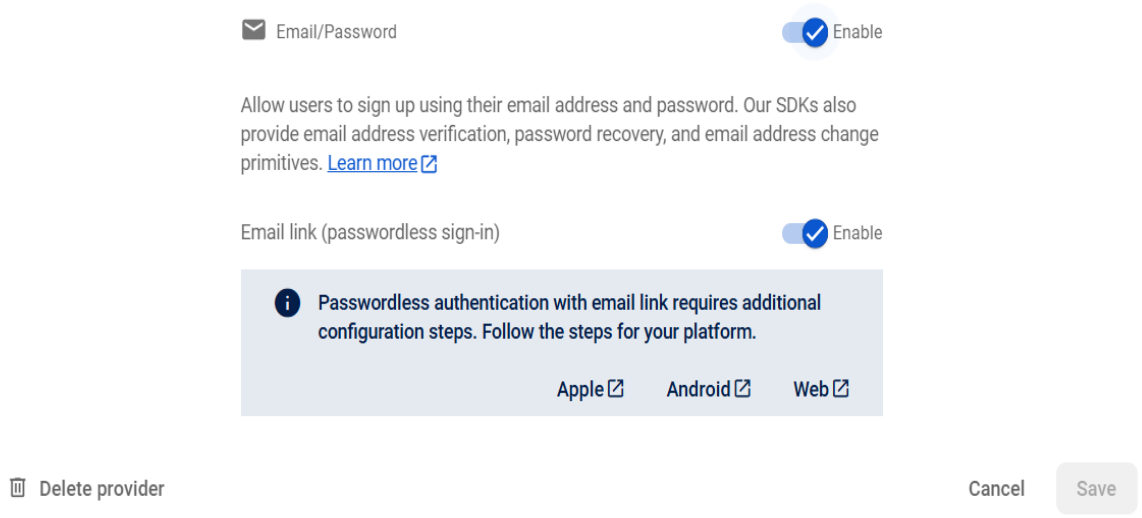
Step 3: - Add Firebase Authentication to Your App

3.1) Add Firebase Authentication Dependencies

```yaml
dependencies:
  flutter:
    sdk: flutter


  firebase_core: ^2.17.0
  firebase_analytics: ^10.5.1
  firebase_auth: ^4.10.1
  cloud_firestore: ^4.9.3


  provider: ^6.0.5


  cupertino_icons: ^1.0.2

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^2.0.0

flutter:
  uses-material-design: true
```

3.2) Enable Authentication in Firebase Console
Go to **Firebase Console → Authentication**.
Click on **Sign-in method** and enable **Email/Password** (or any other method like Google).Click Save





3.3) Implement Authentication in Flutter
Modify main.dart

```
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_auth/firebase_auth.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}
```

Step 4: -Configure Firebase Realtime Database

4.1) Go to Firebase Console → Realtime Database.
4.2) Click **Create Database** → Choose location → Set rules (for development, set read/write to true).
4.3) Click **Publish**.

Code:-

Register_page.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../services/auth_service.dart';

class RegisterScreen extends StatefulWidget {
  const RegisterScreen({super.key});

  @override
  State<RegisterScreen> createState() =>
_RegisterScreenState();
}

class _RegisterScreenState extends
State<RegisterScreen> {
  final _formKey = GlobalKey<FormState>();
  final _nameController =
TextEditingController();
  final _emailController =
TextEditingController();
  final _passwordController =
TextEditingController();
  bool _isLoading = false;
  String? _errorMessage;

  @override
  void dispose() {
    _nameController.dispose();
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.transparent,
        elevation: 0,
        leading: IconButton(
          icon: const Icon(Icons.arrow_back),
          onPressed: () => Navigator.pop(context),
        ),
      ),

      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(24.0),
          child: Form(
            key: _formKey,
            child: Column(
              crossAxisAlignment:
CrossAxisAlignment.start,
              children: [
                const Text(
                  'Create Account',
                  style: TextStyle(
                    color: Colors.white,
                    fontSize: 32,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                const Text(
                  'Join SecureShield VPN',
                  style: TextStyle(
                    color: Colors.white70,
                    fontSize: 16,
                  ),
                ),
                const SizedBox(height: 40),
                // Name Field
                Container(
                  decoration: BoxDecoration(
                    color:
Colors.white.withOpacity(0.1),
                    borderRadius:
BorderRadius.circular(12),
                  ),
                  child: TextFormField(
                    controller: _nameController,
                    style: const TextStyle(color:
Colors.white),
                    decoration: const InputDecoration(
                      hintText: 'Full Name',
                      prefixIcon:
Icon(Icons.person_outline),
```

```
                  border: InputBorder.none,
                  contentPadding:
EdgeInsets.symmetric(horizontal: 20, vertical:
16),
                ),
                validator: (value) {
                  if (value == null || value.isEmpty)
{
                    return 'Please enter your name';
                  }
                  return null;
                },
              ),
            ),
            const SizedBox(height: 16),
            // Email Field
            Container(
              decoration: BoxDecoration(
                color:
Colors.white.withOpacity(0.1),
                borderRadius:
BorderRadius.circular(12),
              ),
              child: TextFormField(
                controller: _emailController,
                style: const TextStyle(color:
Colors.white),
                decoration: const InputDecoration(
                  hintText: 'Email',
                  prefixIcon:
Icon(Icons.email_outlined),
                  border: InputBorder.none,
                  contentPadding:
EdgeInsets.symmetric(horizontal: 20, vertical:
16),
                ),
                validator: (value) {
                  if (value == null || value.isEmpty)
{
                    return 'Please enter your email';
                  }
                  if (!value.contains('@')) {
                    return 'Please enter a valid email';
                  }
                  return null;
                },
              ),
            ),
            const SizedBox(height: 16),
            // Password Field
            Container(
              decoration: BoxDecoration(
                color:
Colors.white.withOpacity(0.1),
                borderRadius:
BorderRadius.circular(12),
              ),
              child: TextFormField(
                controller: _passwordController,
                obscureText: true,
                style: const TextStyle(color:
Colors.white),
                decoration: const InputDecoration(
                  hintText: 'Password',
                  prefixIcon:
Icon(Icons.lock_outline),
                  border: InputBorder.none,
                  contentPadding:
EdgeInsets.symmetric(horizontal: 20, vertical:
16),
                ),
                validator: (value) {
                  if (value == null || value.isEmpty)
{
                    return 'Please enter a password';
                  }
                  if (value.length < 6) {
                    return 'Password must be at least
6 characters';
                  }
                  return null;
                },
              ),
            ),
            const SizedBox(height: 24),
            if (_errorMessage != null)
              Padding(
                padding: const
EdgeInsets.only(bottom: 16),
                child: Text(
                  _errorMessage!,
                  style: TextStyle(color:
Theme.of(context).colorScheme.error),
                  textAlign: TextAlign.center,
```

```
              ),
            ),
            // Register Button
            SizedBox(
              width: double.infinity,
              child: ElevatedButton(
                onPressed: _isLoading ? null :
_register,
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.blue,
                  shape: RoundedRectangleBorder(
                    borderRadius:
BorderRadius.circular(12),
                  ),
                  padding: const
EdgeInsets.symmetric(vertical: 16),
                ),
                child: _isLoading
                    ? const SizedBox(
                  height: 20,
                  width: 20,
                  child: CircularProgressIndicator(
                    color: Colors.white,
                    strokeWidth: 2,
                  ),
                )
                    : const Text(
                  'Create Account',
                  style: TextStyle(
                    fontSize: 16,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
            ),
            const SizedBox(height: 20),
            // Login Option
            Row(
              mainAxisAlignment:
MainAxisAlignment.center,
              children: [
                const Text(
                  'Already have an account? ',
                  style: TextStyle(color:
Colors.white70),
                ),
                TextButton(
                  onPressed: () =>
Navigator.pop(context),
                  child: const Text(
                    'Login',
                    style: TextStyle(
                      color: Colors.white,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
              ],
            ),
          ],
        ),
      ),
    ),
  );
}

Future<void> _register() async {
  print('Register button clicked');

  if (_formKey.currentState!.validate()) {
    setState(() {
      _isLoading = true;
      _errorMessage = null;
    });

    try {
      final authService =
Provider.of<AuthService>(context, listen:
false);

      await authService.signUp(
        _emailController.text,
        _passwordController.text,
        _nameController.text,
      );

      print('Registration successful');

      // No need for manual navigation - the
AuthWrapper will handle it
    } catch (e) {
      print('Registration error: $e');
      setState(() {
```

```
        String message = 'Registration failed';

        if (e.toString().contains('email-already-in-
use')) {
          message = 'This email is already
registered';
        } else if (e.toString().contains('weak-
password')) {
          message = 'Password is too weak';
        } else if (e.toString().contains('invalid-
email')) {
          message = 'Invalid email format';
        }

        _errorMessage = message;
      });
    } finally {
      if (mounted) {
        setState(() {
          _isLoading = false;
        });
      }
    }
  }
}
```

login_page.dart

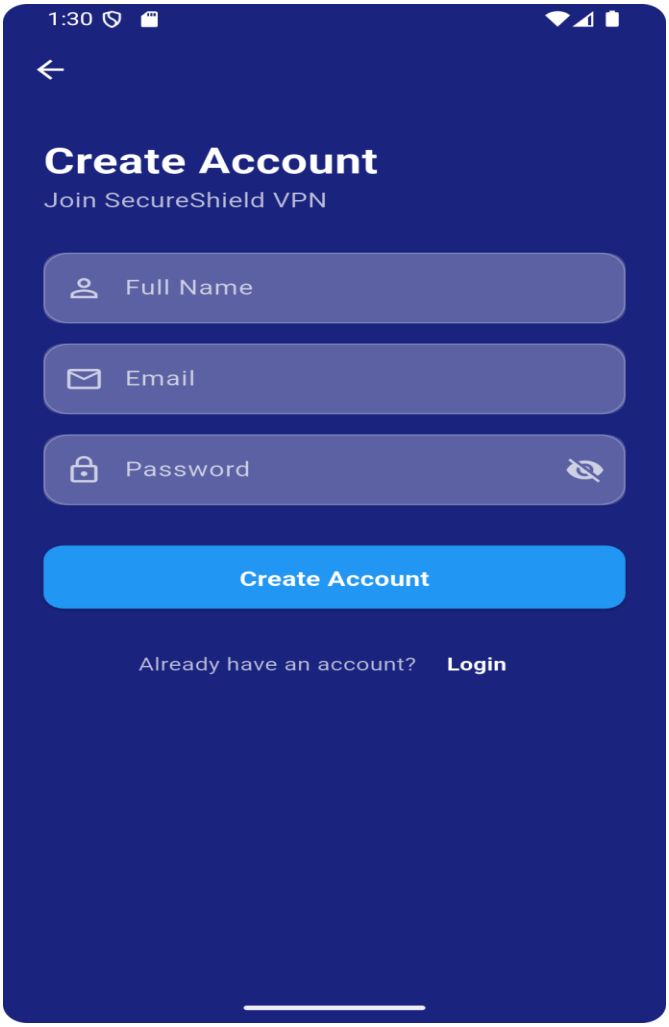
```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'dart:async';
import '../services/vpn_service.dart';

class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});

  @override
  State<HomeScreen> createState() =>
_HomeScreenState();
}

class _HomeScreenState extends
State<HomeScreen> {
  bool isConnected = false;
  String timerText = "00:00:00";
  Timer? _timer;
  int _seconds = 0;
  double downloadSpeed = 0;
  double uploadSpeed = 0;
  int ping = 0;
  String currentServer = "United States";

  @override
  void initState() {
    super.initState();
    print("HomeScreen initialized");

    // Add delay to handle any initialization
    Future.delayed(Duration.zero, () {
      // Get the VPN service
      final vpnService =
Provider.of<VPNService>(context, listen: false);

      // Check if already connected
      setState(() {
        isConnected = vpnService.isConnected;
        print("Initial connection state:
$isConnected");
      });

      // Start timer if connected
      if (isConnected) {
        startTimer();
      }
    });
  }

  void startTimer() {

  print("Starting timer");
    _timer = Timer.periodic(const Duration(seconds:
1), (timer) {
      setState(() {
        _seconds++;
        int hours = _seconds ~/ 3600;
        int minutes = (_seconds % 3600) ~/ 60;
        int seconds = _seconds % 60;
        timerText =
        '${hours.toString().padLeft(2, '0')}:'
            '${minutes.toString().padLeft(2, '0')}:'
            '${seconds.toString().padLeft(2, '0')}';

        downloadSpeed = (300 + (_seconds %
100)).toDouble();
        uploadSpeed = (150 + (_seconds %
50)).toDouble();
        ping = isConnected ? 8 : 0; // Fixed ping to
match UI mockup
      });
    });
  }

  void stopTimer() {
    print("Stopping timer");
    _timer?.cancel();
    setState(() {
      _seconds = 0;
      timerText = "00:00:00";
      downloadSpeed = 0;
      uploadSpeed = 0;
      ping = 0;
    });
  }

  void toggleConnection() async {
    print("Toggle connection button pressed");
    final vpnService =
Provider.of<VPNService>(context, listen: false);

    try {
      if (isConnected) {
        print("Disconnecting VPN");
        // Try to disconnect using Firestore, fallback to
mock method
        try {
          await vpnService.disconnectVPN(dataUsed:
```

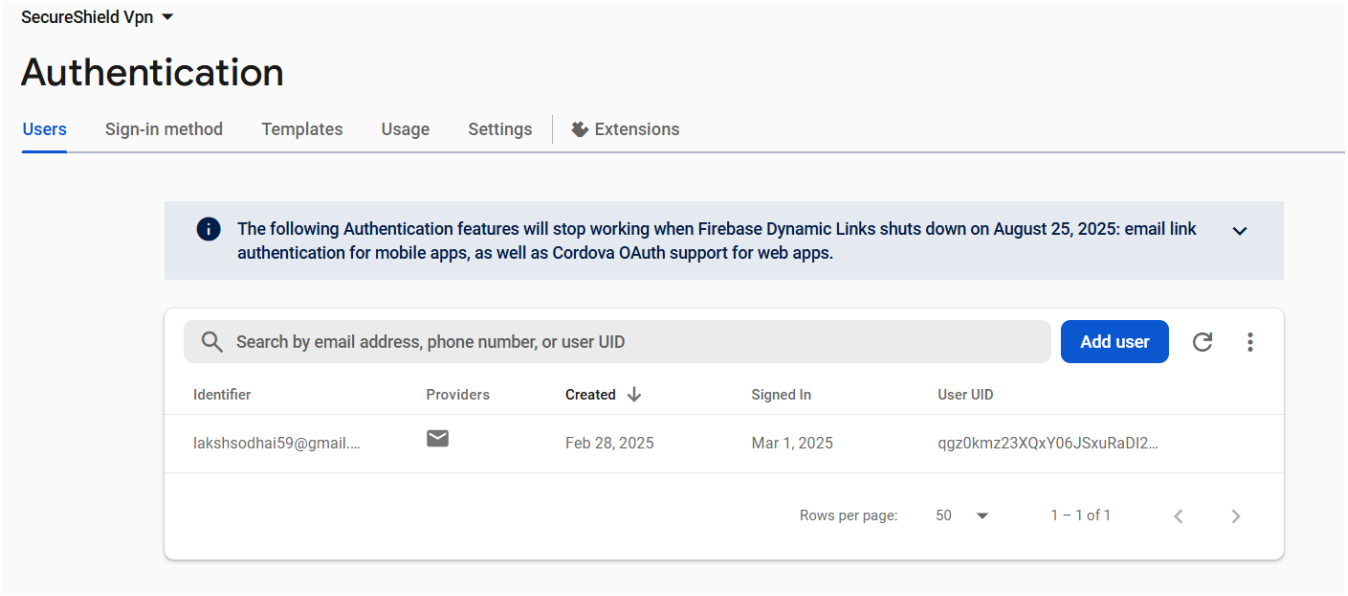
```
downloadSpeed + uploadSpeed);
        } catch (e) {
          print("Using mock disconnect due to
Firestore error: $e");
          await vpnService.mockDisconnect();
        }
        stopTimer();
      } else {
        print("Connecting VPN");
        // Try to connect using Firestore, fallback
to mock method
        try {
          await vpnService.connectVPN(
            serverId: 'us_server',
            serverName: currentServer,
          );
        } catch (e) {
          print("Using mock connect due to
Firestore error: $e");
          await
vpnService.mockConnect(currentServer);
        }
        startTimer();
      }
      setState(() {
        isConnected = !isConnected;
        print("Connection state changed to:
$isConnected");
      });
    } catch (e) {
      print("Error toggling connection: $e");

ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Connection error:
$e")),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    print("Building HomeScreen, isConnected:
$isConnected");

    return Scaffold(
      appBar: AppBar(
        title: const Text('SecureShield VPN'),
        actions: [
          IconButton(
            icon: const
Icon(Icons.notifications_outlined),
            onPressed: () {},
          ),
          IconButton(
            icon: const Icon(Icons.info_outline),
            onPressed: () =>
Navigator.pushNamed(context, '/vpn-details'),
          ),
        ],
      ),
      backgroundColor: Colors.transparent,
      body: SingleChildScrollView(
        child: Padding(
          padding: const
EdgeInsets.symmetric(horizontal: 16),
          child: Column(
            children: [
              const SizedBox(height: 40),
              _buildConnectionButton(),
              const SizedBox(height: 20),
              _buildConnectionStatus(),
              const SizedBox(height: 20),
              _buildTimer(),
              const SizedBox(height: 40),
              _buildLocationAndPing(),
              const SizedBox(height: 40),
              _buildDataUsage(),
              const SizedBox(height: 40),
            ],
          ),
        ),
      ),
    );
  }

  Widget _buildConnectionButton() {
    return InkWell(
      onTap: toggleConnection,
      child: Container(
        width: 200,
        height: 200,
        decoration: BoxDecoration(
          shape: BoxShape.circle,
          color: isConnected ? Colors.green :
Colors.blue,
          boxShadow: [
            BoxShadow(
              color: isConnected ?
Colors.green.withOpacity(0.3) :
Colors.blue.withOpacity(0.3),
```

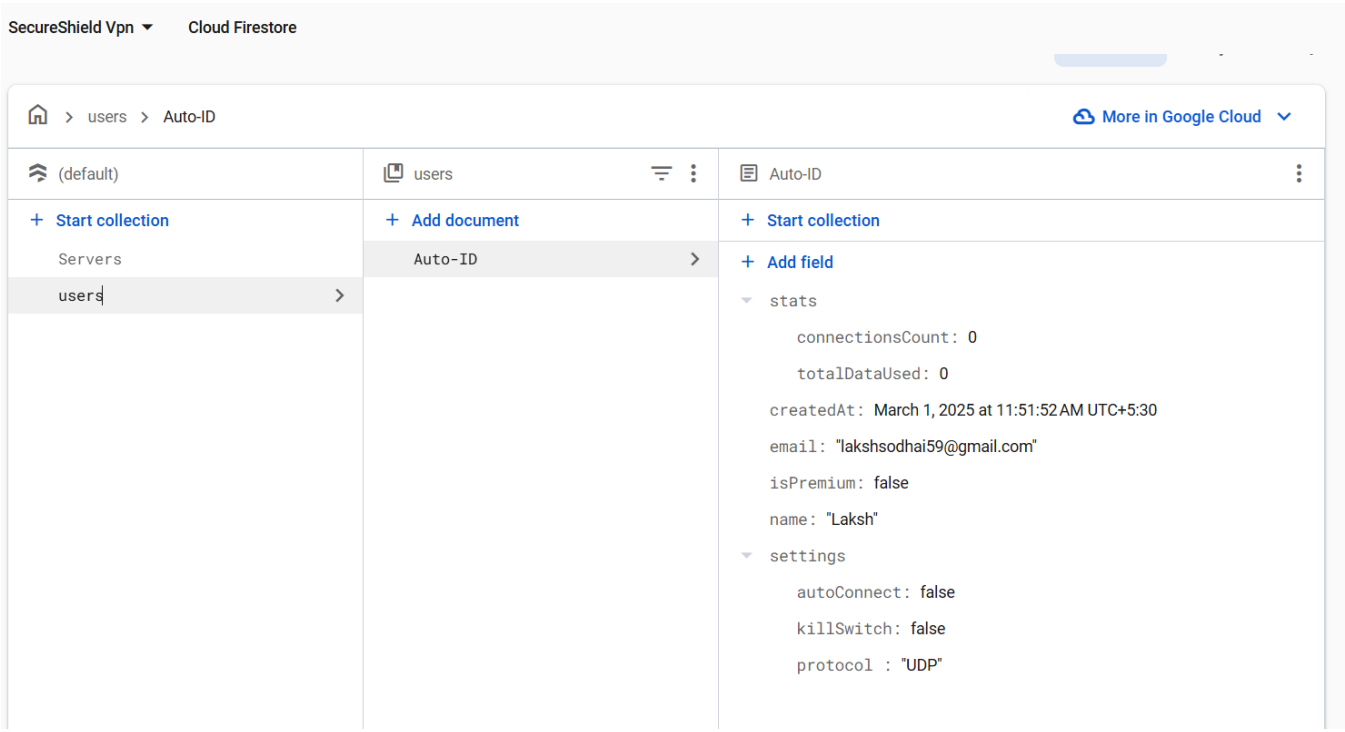
```
  spreadRadius: 10,
            blurRadius: 20,
          ),
        ],
      ),
      child: Column(
        mainAxisAlignment:
MainAxisAlignment.center,
        children: [
          Icon(
            isConnected
                ? Icons.power_settings_new
                : Icons.wifi_off_rounded,
            size: 50,
            color: Colors.white,
          ),
          const SizedBox(height: 8),
          Text(
            isConnected ? 'Connected' : 'Tap to
Connect',
            style: const TextStyle(
              color: Colors.white,
              fontSize: 16,
              fontWeight: FontWeight.bold,
            ),
          ),
        ],
      ),
    ),
  );
 }

 Widget _buildConnectionStatus() {
  return Container(
    padding: const
EdgeInsets.symmetric(horizontal: 30, vertical:
12),
    decoration: BoxDecoration(
      color: isConnected ? Colors.green :
Colors.blue,
      borderRadius: BorderRadius.circular(30),
    ),
    child: Text(
      isConnected ? 'Protected' : 'Not Protected',
      style: const TextStyle(
        color: Colors.white,
        fontWeight: FontWeight.bold,
      ),
    ),
  );
 }

 Widget _buildTimer() {
  return Text(
    timerText,
    style: const TextStyle(
      fontSize: 48,
      fontWeight: FontWeight.bold,
      color: Colors.white,
    ),
  );
 }

 Widget _buildLocationAndPing() {
  return Row(
    mainAxisAlignment:
MainAxisAlignment.center,
    children: [
      Expanded(
        child: Container(
          margin: const EdgeInsets.only(right: 8),
          padding: const
EdgeInsets.symmetric(vertical: 24, horizontal: 16),
          decoration: BoxDecoration(
            color:
Colors.blue.shade800.withOpacity(0.5),
            borderRadius: BorderRadius.circular(15),
          ),
          child: Column(
            children: [
              const Text(
                'us',
                style: TextStyle(fontSize: 30),
              ),
              const SizedBox(height: 8),
              const Text(
                'United States',
                style: TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
              ),
              const Text(
                'FREE',
                style: TextStyle(color: Colors.white70),
              ),
            ],
          ),
        ),
      ),
```

```
Expanded(
        child: Container(
          margin: const EdgeInsets.only(left: 8),
          padding: const
EdgeInsets.symmetric(vertical: 24, horizontal:
16),
          decoration: BoxDecoration(
            color:
Colors.blue.shade800.withOpacity(0.5),
            borderRadius:
BorderRadius.circular(15),
          ),
          child: Column(
            children: [
              Icon(Icons.speed, color:
Colors.orange, size: 30),
              const SizedBox(height: 8),
              Text(
                '$ping ms',
                style: const TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
              ),
              const Text(
                'PING',
                style: TextStyle(color:
Colors.white70),
              ),
            ],
          ),
        ),
      ),
    ],
  );
 }

 Widget _buildDataUsage() {
  return Row(
    mainAxisAlignment:
MainAxisAlignment.center,
    children: [
      Expanded(
        child: Container(
          margin: const EdgeInsets.only(right: 8),
          padding: const EdgeInsets.all(16),
          decoration: BoxDecoration(
            color:
Colors.blue.shade800.withOpacity(0.5),
            borderRadius:
BorderRadius.circular(15),
          ),
          child: Column(
            children: [
              Icon(Icons.download, color:
Colors.green, size: 30),
              const SizedBox(height: 8),
              Text(
                '${downloadSpeed.toStringAsFixed(1)}
KB/s',
                style: const TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
              ),
              const Text(
                'DOWNLOAD',
                style: TextStyle(color: Colors.white70),
              ),
            ],
          ),
        ),
      ),
      Expanded(
        child: Container(
          margin: const EdgeInsets.only(left: 8),
          padding: const EdgeInsets.all(16),
          decoration: BoxDecoration(
            color:
Colors.blue.shade800.withOpacity(0.5),
            borderRadius: BorderRadius.circular(15),
          ),
          child: Column(
            children: [
              Icon(Icons.upload, color: Colors.blue,
size: 30),
              const SizedBox(height: 8),
              Text(
                '${uploadSpeed.toStringAsFixed(1)}
KB/s',
                style: const TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
              ),
              const Text(
                'UPLOAD',
                style: TextStyle(color: Colors.white70),
              ),
            ],
```


1:15
SecureShield VPN
Secure • Fast • Reliable
Email
Password
Forgot Password?
Login
Or continue with
G
Don't have an account? Sign Up


1:30
Create Account
Join SecureShield VPN
Full Name
Email
Password
Create Account
Already have an account? Login


SecureShield Vpn
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.
Search by email address, phone number, or user UID
Add user
Identifier
Providers
Created
Signed In
User UID
lakshsodhai59@gmail....
Feb 28, 2025
Mar 1, 2025
qgz0kmz23XQxY06JSxuRaDI2...
Rows per page:
50
1 – 1 of 1


SecureShield Vpn
Cloud Firestore
users
Auto-ID
More in Google Cloud
(default)
Start collection
Servers
users
users
Add document
Auto-ID
Auto-ID
Start collection
Add field
stats
connectionsCount: 0
totalDataUsed: 0
createdAt: March 1, 2025 at 11:51:52 AM UTC+5:30
email: "lakshsodhai59@gmail.com"
isPremium: false
name: "Laksh"
settings
autoConnect: false
killSwitch: false
protocol : "UDP"