Task 1 - Create a sign-in/sign-up page in Flutter which includes, building the user interface (UI) and implementing the authentication logic.

```
import 'dart:async';
import 'dart:ui';
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

/// Simple in-memory auth model for demo purposes.
/// Replace with your backend (Firebase, REST, Mongo, etc.).
class AuthService {
  static final AuthService _instance = AuthService._internal();
  factory AuthService() => _instance;
  AuthService._internal();

  final Map<String, _User> _users = {
    // demo user
    'demo@demo.com': _User(
      email: 'demo@demo.com',
      name: 'Demo User',
      password: 'demo123',
    ),
  };

  Future<_User> signIn({required String email, required String password}) async {
    await Future.delayed(const Duration(milliseconds: 750));
    final user = _users[email.trim().toLowerCase()];
    if (user == null || user.password != password) {
      throw AuthException('Invalid email or password');
    }
    return user;
  }

  Future<_User> signUp({
    required String name,
    required String email,
    required String password,
  }) async {
    await Future.delayed(const Duration(milliseconds: 750));
    final key = email.trim().toLowerCase();
    if (_users.containsKey(key)) {
      throw AuthException('Account already exists for this email');
    }
```

```
    final user = _User(name: name.trim(), email: key, password: password);
    _users[key] = user;
    return user;
  }
}

class AuthException implements Exception {
  final String message;
  AuthException(this.message);
  @override
  String toString() => message;
}

class _User {
  final String name;
  final String email;
  final String password;
  const _User({required this.name, required this.email, required this.password});
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Neat Auth UI',
      theme: ThemeData(
        brightness: Brightness.light,
        colorScheme: ColorScheme.fromSeed(seedColor: const Color(0xFF6C63FF)),
        inputDecorationTheme: InputDecorationTheme(
          filled: true,
          fillColor: Colors.white.withOpacity(0.08),
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(16),
            borderSide: BorderSide(color: Colors.white.withOpacity(0.2)),
          ),
          enabledBorder: OutlineInputBorder(
            borderRadius: BorderRadius.circular(16),
            borderSide: BorderSide(color: Colors.white.withOpacity(0.15)),
          ),
          focusedBorder: OutlineInputBorder(
            borderRadius: BorderRadius.circular(16),
            borderSide: const BorderSide(color: Color(0xFF6C63FF), width: 1.5),
          ),
          hintStyle: TextStyle(color: Colors.white.withOpacity(0.7)),
          labelStyle: const TextStyle(color: Colors.white),
        ),
      ),
      home: const AuthShell(),
```

```
    );
  }
}

/// Shell that shows a gradient + animated glass card and toggles Login/SignUp
class AuthShell extends StatefulWidget {
  const AuthShell({super.key});

  @override
  State<AuthShell> createState() => _AuthShellState();
}

class _AuthShellState extends State<AuthShell> with SingleTickerProviderStateMixin {
  bool showLogin = true;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        fit: StackFit.expand,
        children: [
          const _AnimatedBackground(),
          Center(
            child: LayoutBuilder(
              builder: (context, constraints) {
                final width = constraints.maxWidth;
                final cardWidth = width < 420 ? width * 0.9 : 400.0;
                return AnimatedSwitcher(
                  duration: const Duration(milliseconds: 300),
                  switchInCurve: Curves.easeOutCubic,
                  switchOutCurve: Curves.easeInCubic,
                  transitionBuilder: (child, anim) => ScaleTransition(scale: anim, child: child),
                  child: _GlassCard(
                    key: ValueKey<bool>(showLogin),
                    width: cardWidth,
                    child: Column(
                      mainAxisSize: MainAxisSize.min,
                      children: [
                        const SizedBox(height: 8),
                        Icon(
                          showLogin ? Icons.lock_outline_rounded :
Icons.person_add_alt_1_rounded,
                          size: 48,
                          color: Colors.white,
                        ),
                        const SizedBox(height: 8),
                        Text(
                          showLogin ? 'Welcome back' : 'Create account',
                          style: const TextStyle(
                            color: Colors.white,
                            fontWeight: FontWeight.w700,
```

```
                    fontSize: 22,
                  ),
                ),
                const SizedBox(height: 6),
                Text(
                  showLogin
                      ? 'Sign in to continue'
                      : 'Sign up to get started',
                  style: TextStyle(color: Colors.white.withOpacity(0.8), fontSize: 13),
                ),
                const SizedBox(height: 20),
                if (showLogin) LoginForm(onToggle: _toggle) else SignUpForm(onToggle:
_toggle),
                const SizedBox(height: 12),
              ],
            ),
          ),
        );
      },
    ),
  ),
 ],
 ),
 );
 }

 void _toggle() => setState(() => showLogin = !showLogin);
}

class _AnimatedBackground extends StatefulWidget {
 const _AnimatedBackground();
 @override
 State<_AnimatedBackground> createState() => _AnimatedBackgroundState();
}

class _AnimatedBackgroundState extends State<_AnimatedBackground>
   with SingleTickerProviderStateMixin {
 late final AnimationController _controller =
     AnimationController(vsync: this, duration: const Duration(seconds: 16))..repeat();

 @override
 void dispose() {
  _controller.dispose();
  super.dispose();
 }

 @override
 Widget build(BuildContext context) {
  return AnimatedBuilder(
   animation: _controller,
   builder: (context, _) {
```

```
        final t = _controller.value;
        return Container(
          decoration: BoxDecoration(
            gradient: LinearGradient(
              begin: Alignment.topLeft,
              end: Alignment.bottomRight,
              colors: [
                Color.lerp(const Color(0xFF2E335A), const Color(0xFF1C1B33), t)!,
                Color.lerp(const Color(0xFF7A77FF), const Color(0xFF9B72F2), 1 - t)!,
              ],
            ),
          ),
          child: CustomPaint(
            painter: _BlobPainter(progress: t),
          ),
        );
      },
    );
  }
}

class _BlobPainter extends CustomPainter {
  final double progress;
  _BlobPainter({required this.progress});
  @override
  void paint(Canvas canvas, Size size) {
    final paint = Paint()..maskFilter = const MaskFilter.blur(BlurStyle.normal, 40);

    paint.color = const Color(0xFF6C63FF).withOpacity(0.35);
    final p1 = Path()
      ..addOval(Rect.fromCircle(
        center: Offset(size.width * (0.2 + 0.1 * progress), size.height * 0.25),
        radius: 120 + 20 * progress,
      ));
    canvas.drawPath(p1, paint);

    paint.color = const Color(0xFF00E5FF).withOpacity(0.28);
    final p2 = Path()
      ..addOval(Rect.fromCircle(
        center: Offset(size.width * (0.85 - 0.1 * progress), size.height * 0.75),
        radius: 140 - 10 * progress,
      ));
    canvas.drawPath(p2, paint);

    paint.color = const Color(0xFFFF6FD8).withOpacity(0.22);
    final p3 = Path()
      ..addOval(Rect.fromCircle(
        center: Offset(size.width * (0.6 + 0.05 * progress), size.height * 0.15),
        radius: 100 + 30 * (1 - progress),
      ));
    canvas.drawPath(p3, paint);
```

```
  }

  @override
  bool shouldRepaint(covariant _BlobPainter oldDelegate) => oldDelegate.progress !=
progress;
}

class _GlassCard extends StatelessWidget {
  final double width;
  final Widget child;
  const _GlassCard({super.key, required this.width, required this.child});

  @override
  Widget build(BuildContext context) {
    return ClipRRect(
      borderRadius: BorderRadius.circular(24),
      child: BackdropFilter(
        filter: ImageFilter.blur(sigmaX: 18, sigmaY: 18),
        child: Container(
          width: width,
          padding: const EdgeInsets.symmetric(horizontal: 18, vertical: 16),
          decoration: BoxDecoration(
            color: Colors.white.withOpacity(0.15),
            borderRadius: BorderRadius.circular(24),
            border: Border.all(color: Colors.white.withOpacity(0.25), width: 1),
            boxShadow: [
              BoxShadow(
                color: Colors.black.withOpacity(0.25),
                blurRadius: 26,
                offset: const Offset(0, 10),
              )
            ],
          ),
          child: child,
        ),
      ),
    );
  }
}

class LoginForm extends StatefulWidget {
  final VoidCallback onToggle;
  const LoginForm({super.key, required this.onToggle});

  @override
  State<LoginForm> createState() => _LoginFormState();
}

class _LoginFormState extends State<LoginForm> {
  final _formKey = GlobalKey<FormState>();
  final _email = TextEditingController(text: 'demo@demo.com');
```

```
final _password = TextEditingController(text: 'demo123');
bool _obscure = true;
bool _loading = false;

@override
void dispose() {
  _email.dispose();
  _password.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Form(
    key: _formKey,
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.stretch,
      children: [
        TextFormField(
          controller: _email,
          style: const TextStyle(color: Colors.white),
          keyboardType: TextInputType.emailAddress,
          decoration: const InputDecoration(
            labelText: 'Email',
            hintText: 'you@example.com',
            prefixIcon: Icon(Icons.alternate_email_rounded, color: Colors.white),
          ),
          validator: (v) {
            if (v == null || v.trim().isEmpty) return 'Email is required';
            final ok = RegExp(r'^[^@\s]+@[^@\s]+\.[^@\s]+$').hasMatch(v.trim());
            return ok ? null : 'Enter a valid email';
          },
        ),
        const SizedBox(height: 12),
        TextFormField(
          controller: _password,
          style: const TextStyle(color: Colors.white),
          obscureText: _obscure,
          decoration: InputDecoration(
            labelText: 'Password',
            hintText: '••••••••',
            prefixIcon: const Icon(Icons.lock_rounded, color: Colors.white),
            suffixIcon: IconButton(
              onPressed: () => setState(() => _obscure = !_obscure),
              icon: Icon(_obscure ? Icons.visibility_rounded : Icons.visibility_off_rounded,
                  color: Colors.white),
            ),
          ),
          validator: (v) => (v == null || v.length < 6) ? 'Min 6 characters' : null,
        ),
        const SizedBox(height: 8),
```

```
        Align(
          alignment: Alignment.centerRight,
          child: TextButton(
            onPressed: () => _showSnack(context, 'Forgot password? (demo)') ,
            child: const Text('Forgot password?', style: TextStyle(color: Colors.white)),
          ),
        ),
        const SizedBox(height: 4),
        ElevatedButton(
          onPressed: _loading ? null : _submit,
          style: ElevatedButton.styleFrom(
            backgroundColor: const Color(0xFF6C63FF),
            foregroundColor: Colors.white,
            shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
            padding: const EdgeInsets.symmetric(vertical: 14),
          ),
          child: _loading
              ? const SizedBox(height: 20, width: 20, child:
CircularProgressIndicator(strokeWidth: 2))
              : const Text('Sign In', style: TextStyle(fontWeight: FontWeight.w700)),
        ),
        const SizedBox(height: 12),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text('No account?', style: TextStyle(color: Colors.white.withOpacity(0.9))),
            TextButton(
              onPressed: widget.onToggle,
              child: const Text('Create one', style: TextStyle(color: Colors.white, fontWeight:
FontWeight.bold)),
            ),
          ],
        ),
      ],
    ),
  );
 }

  Future<void> _submit() async {
    if (!_formKey.currentState!.validate()) return;
    setState(() => _loading = true);
    try {
      final user = await AuthService().signIn(email: _email.text, password: _password.text);
      if (!mounted) return;
      Navigator.of(context).pushReplacement(
        MaterialPageRoute(builder: (_) => HomePage(user: user)),
      );
    } on AuthException catch (e) {
      _showSnack(context, e.message);
    } catch (_) {
      _showSnack(context, 'Something went wrong');
```

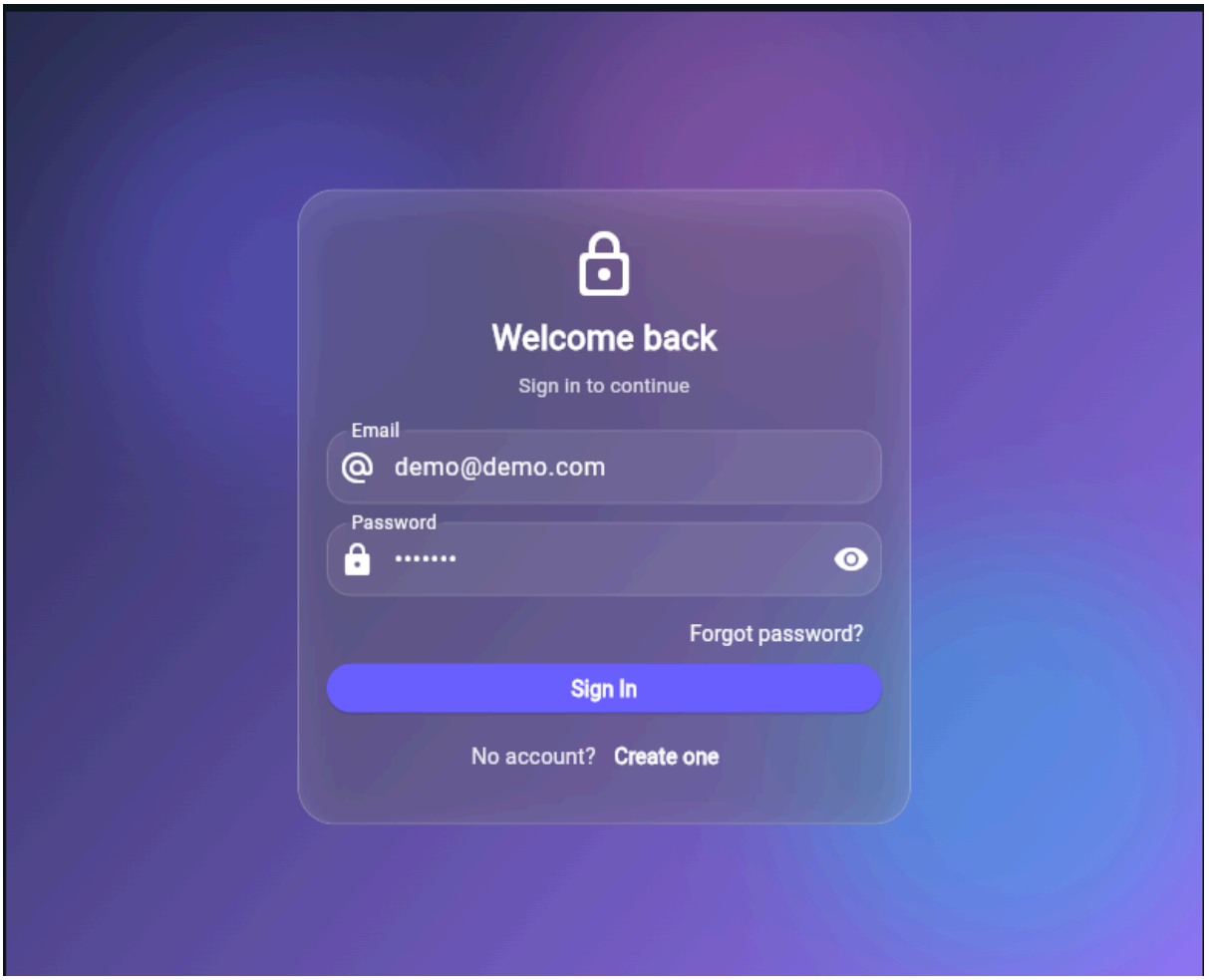
```
    } finally {
      if (mounted) setState(() => _loading = false);
    }
  }
}

class SignUpForm extends StatefulWidget {
  final VoidCallback onToggle;
  const SignUpForm({super.key, required this.onToggle});

  @override
  State<SignUpForm> createState() => _SignUpFormState();
}

class _SignUpFormState extends State<SignUpForm> {
  final _formKey = GlobalKey<FormState>();
  final _name = TextEditingController();
  final _email = TextEditingController();
  final _password = TextEditingController();
  final _confirm = TextEditingController();
  bool _obscure1 = true;
  bool _obscure2 = true;
  bool _loading = false;

  @override
  void dispose() {
    _name.dispose();
    _email.dispose();
    _password.dispose();
    _confirm.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          TextFormField(
            controller: _name,
            style: const TextStyle(color: Colors.white),
            textCapitalization: TextCapitalization.words,
            decoration: const InputDecoration(
              labelText: 'Full name',
              hintText: 'Your name',
              prefixIcon: Icon(Icons.badge_rounded, color: Colors.white),
            ),
            validator: (v) => (v == null || v.trim().length < 2) ? 'Enter your name' : null,
          ),
```

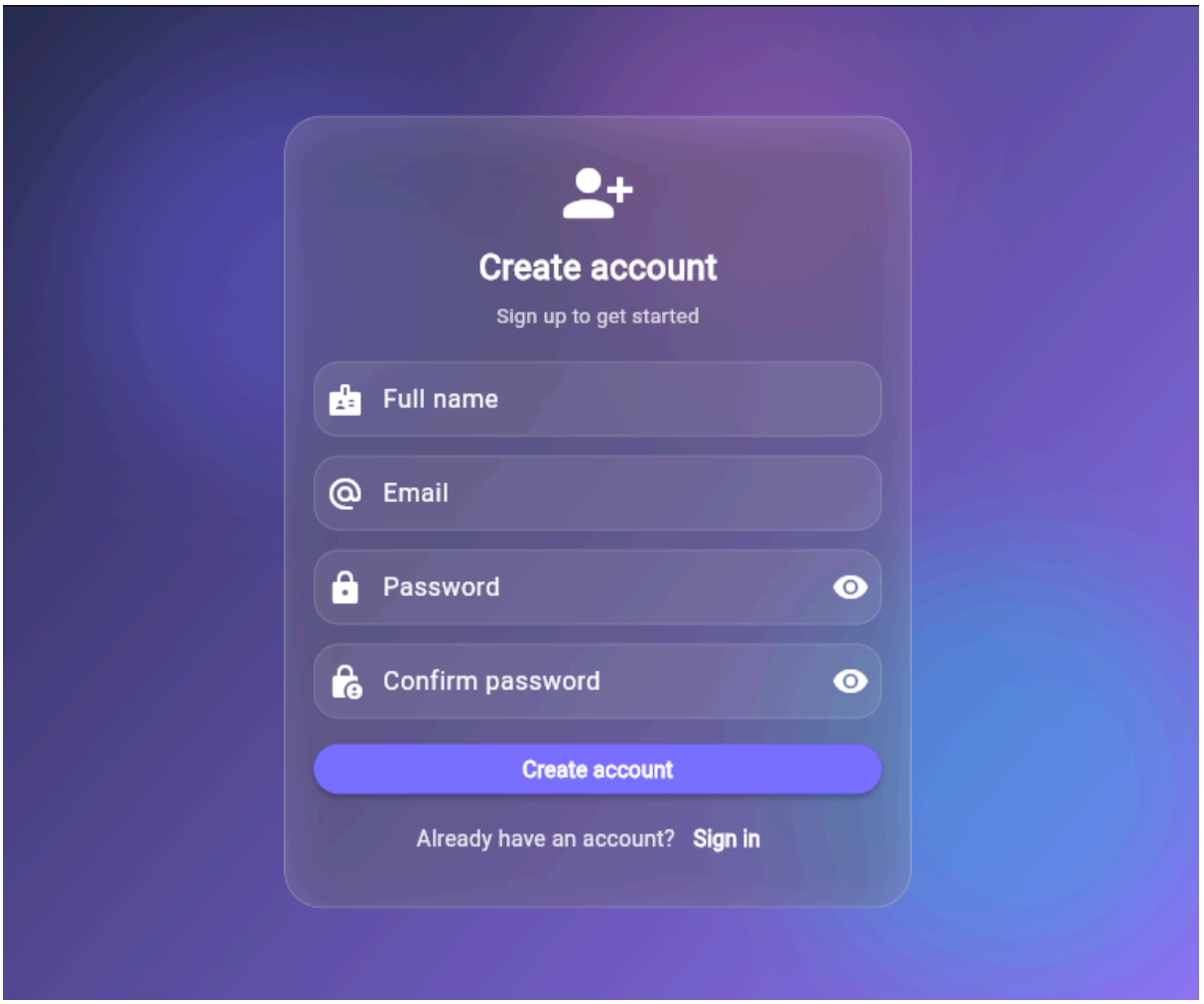
```
const SizedBox(height: 12),
TextFormField(
  controller: _email,
  style: const TextStyle(color: Colors.white),
  keyboardType: TextInputType.emailAddress,
  decoration: const InputDecoration(
    labelText: 'Email',
    hintText: 'you@example.com',
    prefixIcon: Icon(Icons.alternate_email_rounded, color: Colors.white),
  ),
  validator: (v) {
    if (v == null || v.trim().isEmpty) return 'Email is required';
    final ok = RegExp(r'^[^@\s]+@[^@\s]+\.[^@\s]+$').hasMatch(v.trim());
    return ok ? null : 'Enter a valid email';
  },
),
const SizedBox(height: 12),
TextFormField(
  controller: _password,
  style: const TextStyle(color: Colors.white),
  obscureText: _obscure1,
  decoration: InputDecoration(
    labelText: 'Password',
    hintText: 'Min 6 characters',
    prefixIcon: const Icon(Icons.lock_rounded, color: Colors.white),
    suffixIcon: IconButton(
      onPressed: () => setState(() => _obscure1 = !_obscure1),
      icon: Icon(_obscure1 ? Icons.visibility_rounded : Icons.visibility_off_rounded,
          color: Colors.white),
    ),
  ),
  validator: (v) => (v == null || v.length < 6) ? 'Min 6 characters' : null,
),
const SizedBox(height: 12),
TextFormField(
  controller: _confirm,
  style: const TextStyle(color: Colors.white),
  obscureText: _obscure2,
  decoration: InputDecoration(
    labelText: 'Confirm password',
    hintText: 'Re-enter password',
    prefixIcon: const Icon(Icons.lock_person_rounded, color: Colors.white),
    suffixIcon: IconButton(
      onPressed: () => setState(() => _obscure2 = !_obscure2),
      icon: Icon(_obscure2 ? Icons.visibility_rounded : Icons.visibility_off_rounded,
          color: Colors.white),
    ),
  ),
  validator: (v) => (v != _password.text) ? 'Passwords do not match' : null,
),
const SizedBox(height: 16),
```

```
        ElevatedButton(
          onPressed: _loading ? null : _submit,
          style: ElevatedButton.styleFrom(
            backgroundColor: const Color(0xFF6C63FF),
            foregroundColor: Colors.white,
            shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
            padding: const EdgeInsets.symmetric(vertical: 14),
          ),
          child: _loading
              ? const SizedBox(height: 20, width: 20, child:
CircularProgressIndicator(strokeWidth: 2))
              : const Text('Create account', style: TextStyle(fontWeight: FontWeight.w700)),
        ),
        const SizedBox(height: 12),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text('Already have an account?', style: TextStyle(color:
Colors.white.withOpacity(0.9))),
            TextButton(
              onPressed: widget.onToggle,
              child: const Text('Sign in', style: TextStyle(color: Colors.white, fontWeight:
FontWeight.bold)),
            ),
          ],
        ),
      ],
    ),
  );
 }

  Future<void> _submit() async {
    if (!_formKey.currentState!.validate()) return;
    setState(() => _loading = true);
    try {
      final user = await AuthService().signUp(
        name: _name.text,
        email: _email.text,
        password: _password.text,
      );
      if (!mounted) return;
      Navigator.of(context).pushReplacement(
        MaterialPageRoute(builder: (_) => HomePage(user: user)),
      );
    } on AuthException catch (e) {
      _showSnack(context, e.message);
    } catch (_) {
      _showSnack(context, 'Something went wrong');
    } finally {
      if (mounted) setState(() => _loading = false);
    }
```

```
  }
}

class HomePage extends StatelessWidget {
  final _User user;
  const HomePage({super.key, required this.user});

  @override
  Widget build(BuildContext context) {
    final scheme = Theme.of(context).colorScheme;
    return Scaffold(
      appBar: AppBar(
        title: const Text('Home'),
        backgroundColor: scheme.primary,
        foregroundColor: Colors.white,
      ),
      body: Container(
        decoration: const BoxDecoration(
          gradient: LinearGradient(
            begin: Alignment.topLeft,
            end: Alignment.bottomRight,
            colors: [Color(0xFF2E335A), Color(0xFF9B72F2)],
          ),
        ),
        child: Center(
          child: _GlassCard(
            width: 420,
            child: Padding(
              padding: const EdgeInsets.all(8.0),
              child: Column(
                mainAxisSize: MainAxisSize.min,
                children: [
                  const Icon(Icons.celebration_rounded, color: Colors.white, size: 48),
                  const SizedBox(height: 6),
                  Text('Signed in as', style: TextStyle(color: Colors.white.withOpacity(0.9))),
                  const SizedBox(height: 4),
                  Text(user.name, style: const TextStyle(color: Colors.white, fontWeight:
FontWeight.bold, fontSize: 18)),
                  const SizedBox(height: 2),
                  Text(user.email, style: TextStyle(color: Colors.white.withOpacity(0.9))),
                  const SizedBox(height: 16),
                  FilledButton.icon(
                    onPressed: () => Navigator.of(context).pushAndRemoveUntil(
                      MaterialPageRoute(builder: (_) => const AuthShell()),
                      (_) => false,
                    ),
                    icon: const Icon(Icons.logout_rounded),
                    label: const Text('Sign out'),
                    style: FilledButton.styleFrom(
                      backgroundColor: const Color(0xFF6C63FF),
                      foregroundColor: Colors.white,
```

```
                shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(14)),
                padding: const EdgeInsets.symmetric(horizontal: 18, vertical: 12),
              ),
            )
          ],
        ),
      ),
    ),
  ),
 ),
 );
 }
}

void _showSnack(BuildContext context, String msg) {
 ScaffoldMessenger.of(context).showSnackBar(
  SnackBar(content: Text(msg), behavior: SnackBarBehavior.floating),
 );
}
```





Github link:
https://github.com/jayalakshmi-sakthi/Flutter-works