GETTING STARTED WITH FLUTTER – A BEGINNER'S TOOLKIT FOR MOBILE APP DEVELOPMENT

Goal: Build a minimal **Simple Login UI** mobile app using Flutter and Dart that accepts a username and password, validates them locally, and displays a success or error message on login.

1. Quick Summary of the Technology

Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase.
Dart is the programming language used by Flutter.

Real-World Usage: Flutter allows developers to create high-performance, cross-platform apps efficiently.
Example: Alibaba uses Flutter for parts of its Xianyu app to ensure smooth UI across Android and iOS.

2. System Requirements

- **Operating Systems:** Windows 10/11, macOS, Linux
- **Editors/Tools:** VS Code, Android Studio
- **Java Development Kit:** JDK 11 or higher
- **Flutter SDK**
- **Emulator/Simulator:** Android Emulator / iOS Simulator / Desktop support
- **Dependencies:** None for the minimal login UI beyond Flutter itself

---

3. Installation & Setup Instructions

1. Install Flutter SDK from Flutter Installation Guide
2. Add Flutter to your PATH environment variable
3. Run the following in your terminal to check installation:

```
flutter doctor
```

4. Install VS Code and the Flutter/Dart extensions
5. Setup Android Studio and Android SDK, accept licenses:

```
flutter doctor --android-licenses
```

6. Create a new Flutter project:

```
flutter create login_demo
```

7. Run the default Flutter app on an Android emulator:

```
flutter run
```

8. Optional: Enable desktop support:

```
flutter config --enable-windows-desktop

flutter run -d windows
```

---

4. Minimal Working Example

Description:
The Simple Login UI app allows the user to input a username and password. Upon pressing **Login**, a message appears indicating whether the credentials are correct.

main.dart

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

void main() {

 runApp(const LoginDemo());

}

class LoginDemo extends StatelessWidget {

 const LoginDemo({super.key});

 @override

 Widget build(BuildContext context) {

  return MaterialApp(

   home: LoginPage(),

  );

 }

}
```

```
class LoginPage extends StatefulWidget {
  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final _controllerUser = TextEditingController();
  final _controllerPass = TextEditingController();
  String msg = '';

  void validate() {
    setState(() {
      if (_controllerUser.text == 'admin' && _controllerPass.text == '1234') {
        msg = 'Login Successful!';
      } else {
        msg = 'Invalid Credentials';
      }
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Simple Login UI")),
      body: Padding(
        padding: const EdgeInsets.all(20),
        child: Column(
          children: [
            TextField(
              controller: _controllerUser,
              decoration: const InputDecoration(labelText: "Username"),
            ),
            TextField(
              controller: _controllerPass,
              decoration: const InputDecoration(labelText: "Password"),
              obscureText: true,
            ),
            const SizedBox(height: 20),
            ElevatedButton(onPressed: validate, child: const Text("Login")),
            const SizedBox(height: 20),
            Text(msg),
          ],
        ),
      ),
    );
  }
}
```

pubspec.yaml

```
name: login_demo
description: A minimal Flutter login example
```

```
environment:
  sdk: '>=3.0.0 <4.0.0'


dependencies:
  flutter:
    sdk: flutter
```

Expected Output:
The app shows two text fields for username and password, a **Login** button, and a message below that updates upon pressing login.

---

5. AI Prompt Journal

Prompt 1 – README Generation

- **Prompt Used:** Project README Generation
- **Curriculum Link:** Moringa School – Generating and Improving Documentation with AI
- **AI Response Summary:** Provided a structured README template with installation instructions, features, and usage examples.
- **Helpful Part:** Covered all required sections for documentation.
- **Evaluation:** Very helpful; minimal edits needed.

Prompt 2 – Commenting Code

- **Prompt Used:** Add detailed comments to my Flutter login UI code to explain each widget, function, and logic clearly for beginner understanding.
- **AI Response Summary:** Generated inline comments explaining StatefulWidgets, controllers, validation logic, and navigation.
- **Helpful Part:** Makes code beginner-friendly and easier to follow.
- **Evaluation:** Very helpful; ensures peers and instructors can follow code logic.

6. Common Issues & Fixes

- **flutter doctor errors:** Run flutter doctor and fix highlighted issues.

- **Emulator not starting:** Ensure Android Studio setup is correct; restart ADB.
- **PATH not set:** Add Flutter bin folder to system PATH.
- **pubspec.yaml errors:** Check spacing and indentation; run flutter pub get.
- **Dependencies mismatch:** Run flutter pub get.
- **App fails on desktop:** Enable desktop support: flutter config --enable-windows-desktop.
- **Android license issues:** flutter doctor --android-licenses.
- **Hot reload not working:** Restart the app.
- **Dart analyzer errors:** Check import statements.
- **Keyboard obscures text fields:** Wrap Column in SingleChildScrollView.

---

7. References

- Flutter Official Docs
- Dart Language Guide
- Flutter YouTube Tutorials
- StackOverflow Flutter Tag