**Flutter Note-Taking App: Firebase Integration & Development Report**

# 1. Introduction

This report summarizes the experience of building a Flutter note-taking app integrated with Firebase. The app supports full CRUD functionality, email/password and biometric authentication, and uses Bloc for clean architecture and state management. This document outlines the Firebase setup, issues encountered, solutions applied, biometric login integration, Dart analysis, and how all assignment requirements were met.

# 2. Firebase Integration Experience

Connecting Firebase to the Flutter project involved several steps. We used the FlutterFire CLI to register the Android app, download the google-services.json file, and set up dependencies in the Android build files. Authentication and Firestore services were enabled in the Firebase Console.

# 3. Errors and Resolutions

During Firebase initialization on a physical Android device, we encountered a persistent PlatformException error from 'FirebaseCoreHostApi.initializeCore'. The solution involved the following:

* - Ensuring the `minSdkVersion` was set to at least 23
* - Upgrading the NDK version and Gradle settings
* - Rebuilding a minimal test app to isolate the issue

We also faced problems with the biometric login not recognizing previously authenticated sessions. To resolve this, we used the `flutter\_secure\_storage` package to store credentials and log the user in programmatically after successful biometric authentication.

# 4. Biometric Login Integration

Biometric authentication was implemented using the `local\_auth` package and a custom helper class (`LocalAuthHelper`). Users can log in with either email/password or fingerprint at any time. Biometric login does not require a prior session because it reads saved credentials securely from local storage and signs in using Firebase Auth.

# 5. Dart Analyzer Report

To generate a Dart Analyzer report, run the following command in your terminal at the root of the project:

flutter analyze

This command checks for potential code issues, unused imports, type mismatches, and other warnings. We ensured that all analyzer warnings were resolved before final submission.

# 6. Fulfillment of Assignment Requirements

The following assignment requirements have been fulfilled:

* - Clean Architecture: The app separates UI, logic, and services
* - State Management: BLoC is used to manage all state transitions
* - SnackBars: Shown for every success and error case
* - Loader: A loading spinner is shown during login and data fetch
* - Firebase: Integrated with Auth and Firestore
* - Biometric Login: Optional, works independently of session