IN 3510 Wireless Communication & Mobile Networks

Lab 5: Push Notifications in Flutter with Firebase Cloud Messaging (FCM)

Created By Bhanuka Uyanage
Special Credits Rukshan J. Senanayaka

Learning Outcomes

By the end of this lab, you will be able to:

1. Connect a Flutter app to Firebase and enable FCM.
2. Request notification permissions and obtain the device FCM token.
3. Handle notifications in all app states: foreground, background, and terminated.
4. Display local notifications when messages arrive in the foreground.
5. Send test notifications from the Firebase Console and by topic.
6. Navigate to the notification tap & add basic analytics.

Why Push Notifications?

- Engage users when they're not actively using the app
- Drive user retention and re-engagement
- Deliver time-sensitive information
- Personalized user experience
- Important marketing and communication channel

Push Notifications in the real world?

How Duolingo Turned a Free Language App Into a $7.7B Business | WSJ The Economics Of

Types of Notifications

1. Foreground Notifications
   - App is open and visible
   - User actively using the app
   - Can show in-app alerts
2. Background Notifications
   - App minimized but running
   - Shows in the system tray

    - The user can tap to open the app
3. Terminated Notifications
    - App completely closed
    - The system handles delivery
    - Wake the app to handle notifications

Key Components of a Notification

- FCM Token: Unique identifier for each device
- Notification Channel: Groups notifications by importance
- Payload: Data sent in notification
- Handlers: Code that processes notifications

Implementation Steps Summary

1. Firebase setup
    - Add Firebase to the project
    - Configure platforms
    - Get configuration files
2. App Configuration.
    - Request permissions
    - Initialize Firebase
    - Set up handlers
3. Handle Different States.
    - Foreground handler
    - Background handler
    - Tap actions

Implementation Steps Descriptive

1) Create a Firebase Project & Android App

1. Go to **Firebase Console** → Add project → enable Google Analytics (optional).
2. Add Firebase to your Flutter app
    - From any directory, run this command:

```
dart pub global activate flutterfire_cli
```

    - Then, at the root of your Flutter project directory, run this command:

```
flutterfire configure --project=mobile-wireless-1 --platforms=android
```

3. In Firebase → Run → **Cloud Messaging** → make sure it's enabled.

---

2) Add Packages (pubspec.yaml)

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^3.12.0
  firebase_messaging: ^15.1.2
  awesome_notifications_fcm: ^0.10.1
```

Run:

```
flutter pub get
```

---

3) Android Notification Channel & Manifest Tweaks

Android 8.0 and later require a channel for high-priority alerts.

```
<meta-data
android:name="com.google.firebase.messaging.default_notification_channel_id"
      android:value="high_importance_channel" />
<meta-data
      android:name="com.google.firebase.messaging.default_notification_icon"
      android:resource="@mipmap/ic_launcher" />
```

Permissions (Android 13+ needs runtime permission):

```
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
```

> **Note:** We'll request this permission in Dart at runtime.

---

4) Dart Code

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_messaging/firebase_messaging.dart';
import 'package:awesome_notifications/awesome_notifications.dart';
import 'firebase_options.dart';

/// Background handler must be a top-level function
@pragma('vm:entry-point')
Future<void> firebaseMessagingBackgroundHandler(RemoteMessage message) async
{
  await Firebase.initializeApp(options:
DefaultFirebaseOptions.currentPlatform);
}

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(options:
DefaultFirebaseOptions.currentPlatform);

  // Register background handler BEFORE runApp
  FirebaseMessaging.onBackgroundMessage(firebaseMessagingBackgroundHandler);

  // Initialize Awesome Notifications (local)
  await AwesomeNotifications().initialize(
      null, // use default app icon
      [
      NotificationChannel(
        channelKey: 'high_importance_channel',
      channelName: 'High Importance Notifications',
      channelDescription: 'Channel for urgent notifications',
      importance: NotificationImportance.Max,
      defaultRingtoneType: DefaultRingtoneType.Notification,
      ledColor: Colors.white,
      ),
      ],
      debug: true,
  );

  runApp(const MyApp());
}

class MyApp extends StatefulWidget {
```

```dart
  const MyApp({super.key});
  static final GlobalKey<NavigatorState> navigatorKey =
GlobalKey<NavigatorState>();
  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  String? _token;
  String _lastMessage = 'No messages yet';

  @override
  void initState() {
      super.initState();
      _initMessaging();
      // Awesome v0.10.x: use setListeners instead of actionStream
      AwesomeNotifications().setListeners(
      onActionReceivedMethod: (receivedAction) async {
      final payload = receivedAction.payload ?? {};
      final text = payload['message'] ?? 'No payload';
      if (mounted) {
            Navigator.of(context).push(
            MaterialPageRoute(
            builder: (_) => DetailsPage(text: text),
            ),
            );
      }
      },
      );
  }

  Future<void> _initMessaging() async {
      final messaging = FirebaseMessaging.instance;

      // Android 13+/iOS: request permission
      final settings = await messaging.requestPermission(alert: true, badge:
true, sound: true);
      debugPrint('Permission: \${settings.authorizationStatus}');

      // Get token
      _token = await messaging.getToken();
      debugPrint('FCM Token: \$_token');
      setState(() {});
      // Topic subscription (optional)
      await messaging.subscribeToTopic('in3510');

      // Foreground messages: show local notification
      FirebaseMessaging.onMessage.listen((RemoteMessage message) {
      setState(() { _lastMessage = message.data.toString(); });
      final notification = message.notification;
      final title = notification?.title ?? message.data['title'] ?? 'IN3510
Message';
      final body = notification?.body ?? message.data['body'] ?? 'Open to
view details';
      AwesomeNotifications().createNotification(
      content: NotificationContent(
            id: DateTime.now().millisecondsSinceEpoch.remainder(100000),
            channelKey: 'high_importance_channel',
            title: title,
            body: body,
            payload: {
            'route': '/details',
            'message': message.data.toString(),
            },
      ),
      );
      });

      // App opened from background by tapping a system notification
      FirebaseMessaging.onMessageOpenedApp.listen((RemoteMessage message) {
      _handleNotificationTap(message.data);
      });

      // App opened from TERMINATED state via notification
      final initialMessage = await messaging.getInitialMessage();
      if (initialMessage != null) {
      _handleNotificationTap(initialMessage.data);
      }
  }

  void _handleNotificationTap(Map<String, dynamic> data) {
      Navigator.of(context).push(
      MaterialPageRoute(builder: (_) => DetailsPage(text: data.toString())),
      );
  }
  @override
  Widget build(BuildContext context) {
      return MaterialApp(
      title: 'IN3510 FCM Lab',
      navigatorKey: MyApp.navigatorKey,
      home: Scaffold(
      appBar: AppBar(title: const Text('IN3510 • FCM Lab')),
      body: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
            const Text('1) Ensure Firebase is configured.'),
            const SizedBox(height: 8),
            const Text('2) FCM Token:'),
            SelectableText(_token ?? 'Fetching…'),
            const Divider(),
            const Text('Last foreground message data:'),
            Text(_lastMessage),
            const Spacer(),
            ElevatedButton(
                  onPressed: () async {
                  AwesomeNotifications().createNotification(
                  content: NotificationContent(
                        id: 1001,
                        channelKey: 'high_importance_channel',
                        title: 'Local test',
                        body: 'This is a local notification',
                        payload: {'route': '/details', 'message': 'Local test
payload'},
                  ),
                  );
                  },
                  child: const Text('Show Local Test Notification'),
            ),
            ],
            ),
      ),
      ),
      );
  }
}
```

```
class DetailsPage extends StatelessWidget {
  final String text;
  const DetailsPage({super.key, required this.text});
  @override
  Widget build(BuildContext context) {
      return Scaffold(
      appBar: AppBar(title: const Text('Notification Details')),
      body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Text(text),
      ),
      );
  }
}
```

Github Link -
https://github.com/BhanukaC/IN-3510-Wireless-Communication-Mobile-Networks-Lab-5

---

5) Sending Test Messages

5.1 Firebase Console (No code)

- Go to Firebase Console → Messaging → Campaign
- New notification → Target a single device: paste the app's FCM token from the UI.
- Enter Title/Body → Send.

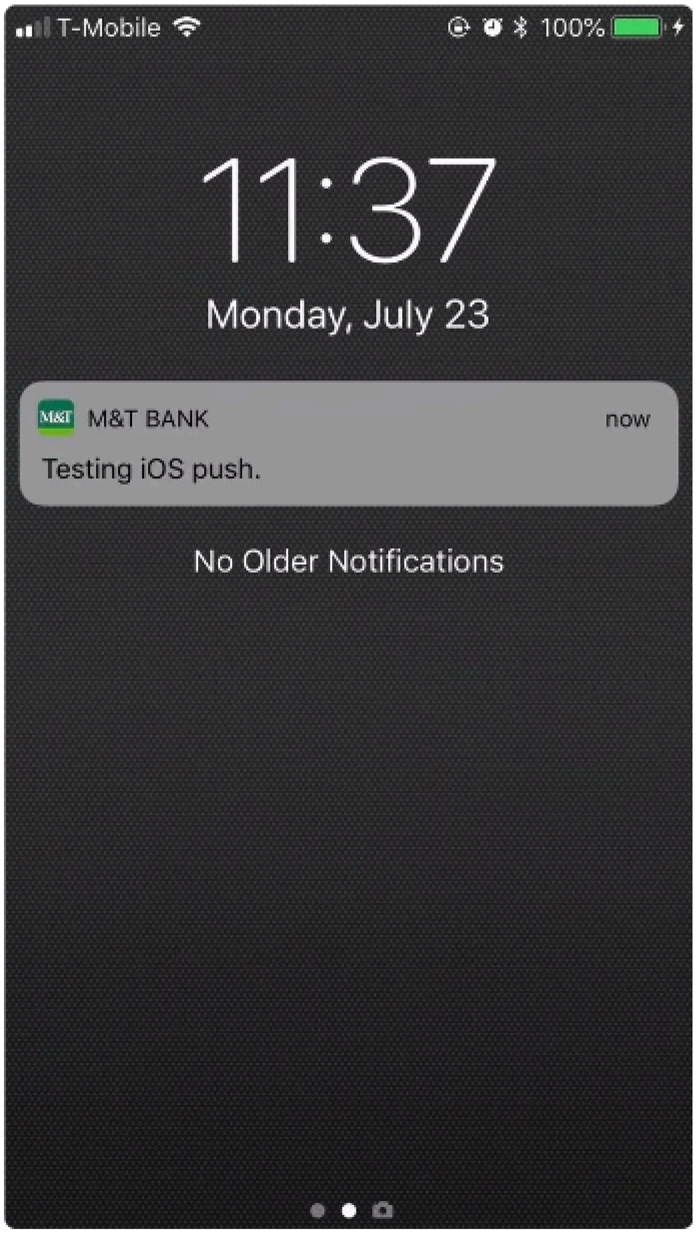
5.2 Topic Messages

- In the console, choose Topic and enter in3510 (we subscribed in code)
- Send to all devices subscribed to in3510.

Best Practices

- Clear and concise messages
- Appropriate timing
- Respect user preferences
- Handle all app states
- Test thoroughly
- Monitor delivery rates

Questions



Testing iOS push.

Readings

[Notifications | FlutterFire]