Sms Scrape

Here:

Change 1: Android Manifest Changes

1. Open the `AndroidManifest.xml` file located at `app/source/main/AndroidManifest.xml`.
2. Add the following lines inside the `<application>` tag to declare required features and permissions:

```
<uses-feature android:name="android.hardware.telephony" andro
id:required="false" />
<uses-permission android:name="android.permission.READ_SMS" /
>
<uses-permission android:name="android.permission.SCHEDULE_EX
ACT_ALARM" />
```

Change 2: Android Build Gradle Changes

1. Open the `build.gradle` file for your app's module.
2. Inside the `repositories` block, add the following Maven repository configuration:

```
gradleCopy code
repositories {
    google()
    mavenCentral()
    maven {
        url "${project(':background_fetch').projectDir}/libs"
    }
}
```

Change 3: main.dart Updates

1. Open the `main.dart` file.
2. Add the following import statement at the top to include the necessary library:

```
import 'package:background_fetch/background_fetch.dart';
@pragma('vm:entry-point')
void backgroundFetchHeadlessTask(HeadlessTask task) async {
  String taskId = task.taskId;
  bool isTimeout = task.timeout;
  if (isTimeout) {
    // This task has exceeded its allowed running-time.  You
must stop what you're doing and immediately .finish(taskId)
    print("[BackgroundFetch] Headless task timed-out: $taskI
d");
    BackgroundFetch.finish(taskId);
    return;
  }
  print("[BackgroundFetch] Headless event received: $taskI
d");
  BackgroundFetch.finish(taskId);
}
```

1. Within the `main()` function, register the background fetch headless task by adding the following line:

```
BackgroundFetch.registerHeadlessTask(backgroundFetchHeadlessT
ask);
```

Change 4: Adding Dependency in pubspec.yaml

1. Open the `pubspec.yaml` file.

2. Add the following line under the `dependencies` section to include the `background_fetch` library:

```
background_fetch: ^1.2.1
```

Change 5: Update in sms_permission.dart

1. Open the `sms_permission.dart` file.
2. Add the following import statement at the top to include the background fetch library:

```
import 'package:background_fetch/background_fetch.dart';
```

1. Add the following variables inside your class declaration:

```
dartCopy code
bool _enabled = true;
int _status = 0;
List<DateTime> _events = [];
```

1. Define the `initPlatformState()` function to configure the background fetch:

```
Future<void> initPlatformState() async {
    // Configure BackgroundFetch.
    int status = await BackgroundFetch.configure(BackgroundFe
tchConfig(
        minimumFetchInterval: 15,
        stopOnTerminate: false,
        enableHeadless: true,
        requiresBatteryNotLow: false,
```

```
        requiresCharging: false,
        requiresStorageNotLow: false,
        requiresDeviceIdle: false,
        requiredNetworkType: NetworkType.NONE
    ), (String taskId) async {  // <-- Event handler
      // This is the fetch-event callback.
      print("[BackgroundFetch] Event received $taskId");

      // Check for permission to read SMS messages
      bool hasPermission = await actions.readSmsPermission();

      if (hasPermission) {
        List<LogsRecord> logs = await queryLogsRecordOnce(
          parent: currentUserReference,
          queryBuilder: (logsRecord) =>
              logsRecord.orderBy('lastRecordTime', descendin
g: true),
          singleRecord: true,
        );
        LogsRecord lastLog = logs.first;
        _model.readMessages = await actions.readMessages(
          lastLog,
        );
        await currentUserReference!.update(createUsersRecordD
ata(
          smsAccess: true,
        ));
        if (_model.readMessages?.length == 0) {
          // No messages found
        } else {
          while (_model.looooop! < _model.readMessages!.lengt
h) {
            _model.transTime = await actions.unixTimestampToD
ateAndTime(
              getJsonField(
                _model.readMessages![_model.looooop!],
```

```
                r'''$.unixTime''',
              ),
            );
            await TransactionsRecord.createDoc(currentUserRef
erence!)
                .set(createTransactionsRecordData(
              dateUnix: getJsonField(
                _model.readMessages?[_model.looooop!],
                r'''$.unixTime''',
              ),
              amount: getJsonField(
                _model.readMessages?[_model.looooop!],
                r'''$.amount''',
              ),
              transacDate: _model.transTime,
            ));
            _model.looooop = _model.looooop! + 1;
          }
          await LogsRecord.createDoc(currentUserReference!)
              .set(createLogsRecordData(
            lastRecordTime: getCurrentTimestamp,
            noOfFields: _model.looooop,
          ));
          _model.looooop = 0;
        }
      } else {
        // Permission denied
      }

      // IMPORTANT:  You must signal completion of your task
or the OS can punish your app
      // for taking too long in the background.
      BackgroundFetch.finish(taskId);
    }, (String taskId) async {  // <-- Task timeout handler.
      // This task has exceeded its allowed running-time.  Yo
u must stop what you're doing and immediately .finish(taskId)
```

```
      print("[BackgroundFetch] TASK TIMEOUT taskId: $taskI
d");
      BackgroundFetch.finish(taskId);
    });
    print('[BackgroundFetch] configure success: $status');

    // If the widget was removed from the tree while the asyn
chronous platform
    // message was in flight, we want to discard the reply ra
ther than calling
    // setState to update our non-existent appearance.
    if (!mounted) return;
  }
```

1. Call the `initPlatformState()` function inside the `initState()` method of your widget:

```
void initState() {
    super.initState();
    _model = createModel(context, () => SmsPermissionModel
());

    initPlatformState();
    WidgetsBinding.instance.addPostFrameCallback((_) => setSt
ate(() {}));
}
```

Remember to replace placeholders like `_model` with your actual variable names and adjust any other context-specific changes.

These changes enhance the functionality of your app, enabling background fetch with specific permissions and configurations. Make sure to test thoroughly after implementing these changes to ensure that everything is working as expected.