Here is the updated README without Google Ads features and with added instructions for setting up Firebase using a demo `firebase_options.dart` and automating the configuration with FlutterFire CLI:

METRO-EASE

Description

This project is a mobile application that offers fare calculation for various routes in Dhaka, Bangladesh. The app allows users to authenticate using Firebase and manage their cards. It calculates fares based on the selected route from a predefined list of destinations.

The app is built using **Flutter** and leverages Firebase for authentication and card management functionalities. The project is inspired by Aniruddha Adhikary, whose open-source project provided a solid foundation for organizing and calculating fare data for MRT routes.

Inspiration

This project is inspired by the work of Aniruddha Adhikary, who originally created the data fetching mechanism for MRT routes. This work has been organized and extended to provide Firebase integration for authentication and card management. The original project is licensed under **GPL-3.0**, and this modified version continues to follow the same licensing terms.

We thank Aniruddha Adhikary for their contribution to open-source development. You can view the original repository here.

Extended Features

- **Firebase Authentication**: Users can sign in using Firebase authentication.
- **Card Management**: Users can manage their cards and view their fare history.
- **Change Card Holder Name**: Users can update the cardholder name.
- **Firebase Firestore Integration**: The app uses Firebase Firestore to store fare data and user information.
- **Android In-App Update**: The app supports Android In-App Update functionality.
- **Cross-Platform Support**: The app is built using Flutter, which allows it to run on both Android and iOS devices.
- **Change Theme**: Users can switch between light and dark themes.
- **Account Deletion**: Users can delete their account and all associated data from the app.
- **User Profile Update**: Users can update their profile information.

To get started with this project, follow the steps below:

Setup Instructions

1. Create a Firebase Project

First, you need to create a Firebase project to enable authentication and card management.

Firebase Authentication Setup

- Go to Firebase Console and create a new Firebase project.
- Add Firebase Authentication to your project and configure the necessary sign-in methods (e.g., email/password, Google sign-in, etc.).
- Enable Firebase Firestore for storing user and card data.

Firebase Firestore Setup

Create a Firestore collection for storing fare information. In your Firestore database, create a collection `fare` with a document `fare`, and populate it with the following structure:

```
fare: {
  "Agargaon": 60,
  "Bangladesh Secretariat": 90,
  "Bijay Sarani": 60,
  "Dhaka University": 90,
  "Farmgate": 70,
  "Kamlapur": 100,
  "Karwan Bazar": 80,
  "Kazipara": 40,
  "Mirpur 10": 40,
  "Mirpur 11": 30,
  "Motijheel": 100,
  "Pallabi": 30,
  "Shahbag": 80,
  "Shewrapar": 50,
  "Uttara Center": 20,
  "Uttara North": 0,
  "Uttara South": 20
}
```

2. Clone the Repository

To start working with the project, clone the repository to your local machine:

```
git clone https://github.com/Piistech/MetroEase.git
```

3. Install Dependencies

Navigate to the project directory and install the necessary dependencies:

```
cd MetroEase
flutter pub get
```

4. Configure Firebase in Your Flutter Project

Manual Configuration

- Download the Firebase configuration files for both Android and iOS from the Firebase Console:
  - For Android: `google-services.json` -> Place it in the `android/app` directory.
  - For iOS: `GoogleService-Info.plist` -> Place it in the `ios/Runner` directory.

Automate with FlutterFire CLI

Alternatively, use the FlutterFire CLI to generate the necessary `firebase_options.dart` file:

1. Install the Firebase CLI:
   Download and install Firebase CLI.
2. Add the FlutterFire CLI:

   ```
   dart pub global activate flutterfire_cli
   ```

3. Configure the project with Firebase:
   Run the following command and follow the prompts to configure your app:

   ```
   flutterfire configure
   ```

4. The `firebase_options.dart` file will be generated automatically in the `lib` folder. Import this file in your project where Firebase initialization is needed.

Example `firebase_options.dart`:

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

const FirebaseOptions firebaseOptions = FirebaseOptions(
  apiKey: "YOUR_API_KEY",
  authDomain: "YOUR_PROJECT_ID.firebaseapp.com",
  projectId: "YOUR_PROJECT_ID",
  storageBucket: "YOUR_PROJECT_ID.appspot.com",
  messagingSenderId: "YOUR_SENDER_ID",
  appId: "YOUR_APP_ID",
  measurementId: "YOUR_MEASUREMENT_ID",
);
```

5. Run the Project

Now, you are ready to run the project. You can launch the app on your preferred device using the following command:

```
flutter run
```

6. Modify Firebase Authentication Rules (Optional)

Ensure that your Firestore has proper rules for secure read and write access to the data.

Example Firestore rules:

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

7. Android In-App Update Integration

This project includes Android In-App Update functionality. No additional setup is needed.

License

This project is licensed under the **GPL-3.0 License**. See the LICENSE file for more details.

Acknowledgements

- This project is inspired by the work of **Aniruddha Adhikary**, whose original project provided the core fare calculation functionality.
- Thanks to **PIISTECH LTD** for contributing to and extending this project.