

AidMap: Developing a Civic Accessibility Application with Flutter and Huawei Mobile Services*

Arkadiusz NIEDZIELA¹, Katarzyna KOSIAK² and Piotr ZWIERZYKOWSKI³

¹Faculty of Computing and Telecommunications, Poznan University of Technology, Poznan, Poland

²Sustainables Student Research Group, Faculty of Civil and Transport Engineering, Poznan University of Technology, Poznan, Poland

³Institute of Communication and Computer Networks, Faculty of Computing and Telecommunications, Poznan University of Technology, Poznan, Poland

Correspondence should be addressed to: Arkadiusz NIEDZIELA, arkadiusz.niedziela@student.put.poznan.pl

*Presented at the 47th IBIMA International Conference, 29-30 June 2026, Madrid, Spain

Abstract

Urban accessibility remains an underdeveloped aspect of urban planning, with public spaces often inaccessible to people with disabilities. Since 2019, new Huawei smartphones have shipped without Google Mobile Services (GMS), complicating application development for a substantial share of the Android market. This paper reports on AidMap, a civic-technology mobile application for reporting accessibility barriers, built with Flutter and Huawei Mobile Services (HMS) and intentionally independent of GMS. We document the architectural decisions and integration trade-offs encountered during development and identify a structural gap in the HMS ecosystem: no augmented reality (AR) solution is currently available that is both GMS-independent and not constrained to Kirin-based hardware. The findings are intended to inform developers evaluating HMS-only or hybrid GMS/HMS strategies.

Keywords: Huawei Mobile Services, Flutter, mobile development, urban accessibility

Introduction

Urban planning rarely treats accessibility as a primary concern. The World Health Organization estimates that around 1.3 billion people live with some form of disability — more than 16% of the global population (World Health Organization 2023). In Poland, a review by the Supreme Audit Office (NIK) covering 24 municipalities found accessibility issues in every public facility it inspected; not one of the 94 sites passed as fully accessible (Najwyższa Izba Kontroli 2018). The numbers point to a systemic gap rather than a handful of isolated oversights.

A parallel trend complicates how such issues can be reported on mobile devices. Since 2019, new Huawei phones have shipped without Google Mobile Services (GMS), following the company's placement on the U.S. Entity List (Bureau of Industry and Security, U.S. Department of Commerce 2019; Reuters 2019). Apps that rely on Google APIs for maps, location, or push notifications therefore do not work out of the box on a substantial portion of newer devices in markets where Huawei still holds share.

This paper describes AidMap, a mobile application for reporting urban accessibility barriers that targets exactly that gap. AidMap is built in Flutter for cross-platform reach and uses Huawei Mobile Services (HMS) as its service layer, making it GMS-independent by design. We do not present AidMap as a finished product. The

paper instead focuses on the technical decisions made during development, the friction encountered while integrating HMS, and the trade-offs that came with those choices.

Technological Background

The GMS Dependency Problem

Google Mobile Services (GMS) is a proprietary stack of APIs and system services that an Android device needs in order to plug into the Google ecosystem (Google Developers 2026): authentication, location, push notifications, and distribution through Google Play. After the U.S. export restrictions were imposed on Huawei in May 2019 (Bureau of Industry and Security, U.S. Department of Commerce 2019), every Huawei phone released from mid-2019 onward has shipped without it. A GMS-dependent application simply will not run on this hardware.

Huawei's response was to build a parallel stack of its own. Huawei Mobile Services (HMS) covers the same ground as GMS, with HMS Core providing the SDKs (Huawei calls them "kits") and AppGallery handling distribution (Huawei Developers 2026).

Developers who want to cover both ecosystems usually pick between two strategies. One is a single build that checks for available services at runtime via `HuaweiApiAvailability` and `GoogleApiAvailability`, and selects the right implementation accordingly. The other relies on Android product flavors (Android Developers 2026) to compile separate variants for each ecosystem from a shared codebase, with platform-specific source sets; this is also what Huawei recommends in its official developer training (Huawei Developer Academy 2024). AidMap follows a stripped-down version of the second approach. We build only the HMS variant, since the project context and our explicit focus on the Huawei ecosystem made supporting GMS unnecessary.

Flutter as a Cross-Platform Framework

Flutter is an open-source cross-platform framework from Google (Flutter 2026). It compiles a single Dart codebase to Android, iOS, web, and desktop targets, ships its own rendering engine rather than binding to native UI components, reaches performance close to native, and offers hot reload along with a sizable third-party package ecosystem.

Our reason for choosing it, however, was narrower. The HMS Core team publishes an official set of Flutter plugins (HMS Core Team 2026) that expose the kits we needed (Map Kit, Location Kit, Account Kit, and several others) through a Dart API, which meant no native Android code on our side for any of the HMS integration. A secondary benefit was that the same codebase could later be extended to iOS without forking the project — something that mattered with only three people on the team.

System design and HMS integration

System Architecture

AidMap follows a standard client-server architecture. The client is a Flutter mobile application that talks to a Flask backend over a REST API; the backend runs on a private VPS, and the app itself is distributed through Huawei AppGallery. We also prototyped a DeepSeek-powered conversational interface, which was deployed but is not evaluated in this paper. The original design called for Huawei Cloud services (OBS, TaurusDB) as the production target, but the prototype documented here still runs on the VPS.

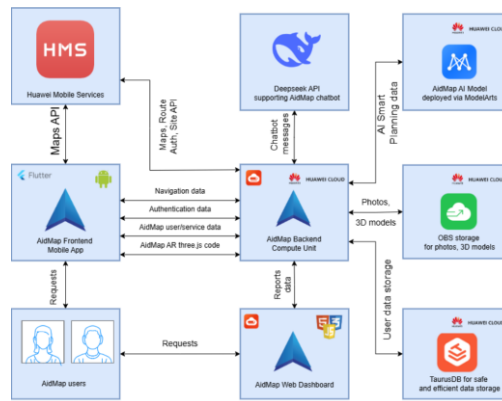


Fig 1. High-level architecture of the AidMap system.

On the device side, the HMS layer sits between the Flutter application and the underlying system services. Each kit we use communicates with the corresponding HMS Core service installed on the phone, taking the place a Google services dependency would otherwise occupy. Figure 1 shows the overall layout, including the components that remained on the roadmap.

Deployed HMS Kits

We selected HMS kits according to AidMap’s functional requirements. Each one below is listed with its GMS counterpart for reference.

Map Kit replaces the Google Maps SDK and forms the core of the map view: it draws the base map, renders pins for reported barriers, and filters them by category. Integration with Flutter goes through Huawei’s official `huawei_map` plugin.

Location Kit stands in for Google Location Services. We use it to fetch the user’s current coordinates when they file a new report, to display their position on the map, and as the location source for the AR module.

Navi Kit replaces the Google Directions API and provides turn-by-turn navigation. We extended its use slightly: AidMap also surfaces the accessibility barriers that fall along a computed route.

Site Kit replaces the Google Places API and powers the in-app search for places and addresses.

Account Kit replaces Google Sign-In. Users authenticate with their Huawei ID, so there is no separate AidMap account to create.

AppGallery Connect takes Firebase’s role as the backend services platform, covering analytics and distribution through AppGallery.

Push Kit, the HMS counterpart to Firebase Cloud Messaging, was not integrated — it was deprioritised due to time constraints.

The Augmented Reality Gap

AidMap was intended to include an AR view that overlaid accessibility barriers onto the camera feed. On the development device — a Huawei Nova 12s (model FOA-LX9) with a Qualcomm Snapdragon 778G — none of the off-the-shelf options worked, and we evaluated four alternatives before abandoning the feature. **ARCore**, the default AR stack on Android, is out by construction: it requires GMS (Google Developers, n.d.), which Huawei devices do not have.

AREngine, Huawei’s own AR engine, only runs on devices with a Kirin SoC (Huawei Developers, n.d.). The Nova 12s used in this work is Snapdragon-based, and tying the application to a specific processor family would have undercut the hardware-independence goal anyway.

AR.js looked promising at first, since a Three.js-based web library should in principle sidestep Google’s tooling. In practice it still needs a WebView to host the camera view, and this is where the ecosystem split shows: Huawei devices ship a proprietary WebView (`com.huawei.webview`) in place of Android System WebView (`com.google.android.webview`), which on standard Android is delivered through the Play Store as

part of the Google services stack. The Huawei WebView did not load AR.js's WebGL and camera pipeline in our tests, and the Google WebView was simply not available on the device.

The remaining option was a custom implementation on plain **Three.js**. We started building one, but synchronising the IMU readings with GPS and the real-time render loop turned out to be more than a prototype could absorb, and the feature was deferred.

There is a significant gap: on a Huawei device today, there is no AR solution that is both HMS-independent and not locked to a specific processor family. For anyone trying to build an AR application outside the Kirin lineup, this constitutes a structural limitation.

Discussion

What Worked Well

Integrating the core HMS kits into Flutter went smoothly, mainly because the HMS Core team maintains official plugins for them. Map Kit, Location Kit, Site Kit, and Account Kit all expose a Dart API that closely tracks their GMS equivalents, so the learning curve was short and a three-person student team could deliver a working prototype within the project timeframe. The finished application was published on AppGallery and operated correctly on Huawei devices without GMS, which was the goal.

Flutter proved well-suited to the HMS context for a second reason beyond plugin availability: its architecture. Because Flutter renders through its own engine rather than wrapping native UI widgets, most of the UI code behaves the same way regardless of whether the device runs GMS or HMS underneath. We could concentrate the integration work on the handful of places where the app actually touches system services, instead of reworking the entire presentation layer.

Limitations and Challenges

The sharpest limitation we encountered was the missing AR feature. As described in Section 3, none of the libraries we tested satisfied both GMS independence and broad hardware compatibility at the same time. There is a gap in the HMS ecosystem here, and it rules out a category of applications — AR-driven ones in particular — unless the developer is willing to lock onto Kirin-based devices.

Distribution reach is the other obvious constraint. Outside China, AppGallery's user base is still a small fraction of Google Play's, so an HMS-only application has a smaller potential audience. For a tool with a social purpose, like one for reporting accessibility barriers, that ceiling matters.

Documentation and community resources were a third source of friction. The HMS ecosystem has a much smaller pool of Stack Overflow answers, sample projects, and blog posts than GMS and Firebase. In day-to-day work this meant leaning more heavily on Huawei's official documentation, which is generally thinner than Google's equivalent, and spending more time diagnosing integration issues than we would have expected on the GMS side.

One more caveat: this prototype never reached production. The data shown in the application is synthetic, and the planned migration to Huawei Cloud infrastructure did not happen. The results should be read as a prototype rather than as a deployed system.

Implications for Developers

A few practical observations come out of the AidMap experience for anyone considering HMS.

An HMS-only approach is justifiable in a narrow set of situations: when the target market has a high concentration of Huawei devices, when the project is a proof of concept or research effort oriented toward the Huawei ecosystem in the first place, and when the functional requirements do not run into the limitations we encountered (AR being the obvious one). Outside those cases, a single build with runtime service detection looks like the better default, since it keeps the application reachable on the full Android device population.

One final qualifier. What we describe here reflects the state of the HMS ecosystem as of mid-2025. Huawei's platform is moving, HarmonyOS NEXT in particular, and the tools and trade-offs we discuss may shift accordingly.

Conclusions

AidMap shows that a working civic-tech mobile application can be built on top of HMS and Flutter without any dependency on GMS. The core features — maps, location, authentication, and route planning — all came together on top of the official HMS Flutter plugins, and the finished application was released on AppGallery and operated on Huawei devices that have no Google services at all.

The single feature we were unable to deliver was augmented reality. As detailed in Section 3, there is currently no AR option for Huawei hardware that avoids both GMS and a hard tie to Kirin processors, which leaves any AR-driven application either compromised or impossible to build under the constraints we worked with.

Three directions follow naturally from here. The first is moving to a single-build setup with runtime service detection, so the application runs on both HMS and GMS devices from one codebase. The second is field testing with real users in an actual urban environment, which the prototype has not yet seen. The third is completing the migration to Huawei Cloud infrastructure that was originally planned for production.

Acknowledgments

This research was funded in part by the Polish Ministry of Science and Higher Education (grant SBAD-MK 0313/SBAD/1314).

- Android Developers (2026) 'Configure build variants,' *Android Developers*. [Online], [Retrieved 10 May 2026], <https://developer.android.com/build/build-variants>
- Bureau of Industry and Security, U.S. Department of Commerce (2019) 'Addition of entities to the entity list,' *Federal Register*. [Online], [Retrieved 10 May 2026], <https://www.federalregister.gov/documents/2019/05/21/2019-10616/addition-of-entities-to-the-entity-list>
- Flutter (2026) 'Flutter documentation,' *Flutter*. [Online], [Retrieved 10 May 2026], <https://docs.flutter.dev/>
- Google Developers (2026) 'Overview of Google Play services,' *Google Developers*. [Online], [Retrieved 10 May 2026], <https://developers.google.com/android/guides/overview>
- Google Developers (n.d.) 'ARCore supported devices,' *Google Developers*. [Online], [Retrieved 10 May 2026], <https://developers.google.com/ar/devices>
- HMS Core Team (2026) 'HMS Flutter plugins,' *GitHub*. [Online], [Retrieved 10 May 2026], <https://github.com/HMS-Core/hms-flutter-plugin>
- Huawei Developer Academy (2024) 'Introduction to GMS + HMS using product flavor,' *Huawei Developer Academy*. [Online], [Retrieved 10 May 2026], <https://developer.huawei.com/consumer/en/training/course/video/C101638943305969006>
- Huawei Developers (2026) 'HMS Core,' *Huawei Developers*. [Online], [Retrieved 10 May 2026], <https://developer.huawei.com/consumer/en/hms/>
- Huawei Developers (n.d.) 'HUAWEI AR Engine – HMS Core,' *Huawei Developers*. [Online], [Retrieved 10 May 2026], <https://developer.huawei.com/consumer/en/hms/huawei-arengine/>
- Najwyższa Izba Kontroli (2018) 'Dostępność przestrzeni publicznej dla osób starszych i niepełnosprawnych,' *Najwyższa Izba Kontroli*. [Online], [Retrieved 10 May 2026], <https://www.nik.gov.pl/plik/id,18229,vp,20827.pdf>
- Reuters (2019) 'Exclusive: Google suspends some business with Huawei after Trump blacklist,' *Reuters*. [Online], [Retrieved 10 May 2026], <https://www.reuters.com/article/world/exclusive-google-suspends-some-business-with-huawei-after-trump-blacklist-sou-idUSKCN1SP0NC/>
- World Health Organization (2023) 'Disability,' *World Health Organization*. [Online], [Retrieved 10 May 2026], <https://www.who.int/news-room/fact-sheets/detail/disability-and-health>