Follow Me Live: A Real-Time Person-to-Person Tracking Prototype on Google Maps

Adhitya Muthukumar
Third-year Undergraduate Student, Dept. of School of Computing Engineering, KIIT Bhubaneswar
adhitya.muthukumar23@gmail.com

ABSTRACT
This work introduces a fully web-based prototype that achieves true real-time, person-to-person location following on Google Maps, addressing a critical gap in existing "share live location → get directions" workflows in applications such as Google Maps and WhatsApp where navigation relies on periodic **static snapshots rather than continuous positional updates**. The proposed system enables two users such as a rider in one vehicle and a follower in another to visualize each other as smoothly moving, continuously updated markers on a shared map interface. **Simultaneously, the system provides dynamically refreshing route guidance, distance estimates, and traffic-aware ETAs** for both parties, replicating the behaviour of high-end ride-hailing platforms.
The prototype is implemented as a lightweight Progressive Web App that requires no native installation. It leverages **HTML5 Geolocation** to obtain device-level GPS readings, **Firebase Realtime** Database for low-latency bidirectional synchronization of location data, and the **Google Maps JavaScript and Directions APIs** to render real-time marker animation, navigation style camera tracking, and automatic route recomputation. The architecture is entirely serverless, operates within free tier cloud limits, and is optimized for rapid deployment, student research, and experimental prototyping.

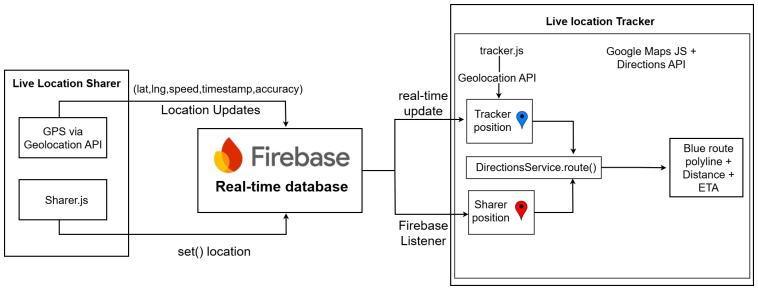
Key words: Dynamic live tracking on GMaps, Live location sharing, Person-to-person navigation

For the live demo, visit prototype web app.
For sharing live location sharer live location page
Video link: Prototype demo video
The source code is available on GitHub: project repository.

1 INTRODUCTION

Live location sharing has become a standard feature in modern messaging and navigation applications, yet most existing implementations exhibit static behaviour when paired with route guidance. When a user selects "Get directions" from a shared live location, Google Maps typically computes a route to the sender's current coordinates and then treats that point as a fixed destination even as the sender continues to move. This creates a misleading sense of real-time tracking while failing to dynamically update the route, distance, or estimated time of arrival (ETA). Such limitations become particularly evident in practical scenarios, for example when one person follows a friend's cab to the airport: both users are travelling through variable traffic conditions, yet the follower receives only a one-time route snapshot rather than continuously adapting navigation cues. In contrast, commercial ride-hailing platforms such as Uber and Ola clearly demonstrate that high-resolution, real-time positional updates with responsive ETAs are technically achievable. However, these capabilities remain inaccessible to ordinary users within standard map applications, which still do not dynamically update navigation in response to the movement of both parties.

This project directly addresses that functional gap by developing a browser-based prototype that transforms a shared live location into a fully dynamic, moving destination. The system enables a "sharer" and "follower" to view each other as continuously



updating markers on a shared Google Map, with the follower's route, distance, and ETA recomputed in real time as both parties move. The prototype integrates HTML5 Geolocation for device-level GPS acquisition, Firebase Realtime Database for low-latency synchronization of positional data, and the Google Maps JavaScript and Directions APIs to deliver seamless marker animation, navigation style camera behaviour, and traffic aware rerouting.

The architecture is entirely serverless, operating exclusively on free tier cloud resources and standard mobile browsers, which makes the system both low-cost and highly accessible. This design enables students, researchers, and developers to experiment with true real-time location interaction on the web without the overhead of native app development or custom backend infrastructure. By demonstrating that continuous person-to-person navigation is feasible using only lightweight web technologies, the prototype closes a practical gap in current location-sharing tools and opens new directions for research in collaborative mobility, safety monitoring, and real-time geo-synchronous applications. A long-term objective of this work is to enable the integration of such true two-party dynamic tracking capabilities into Google Maps.

2 PROTOTYPE ARCHITECTURE OVERVIEW

The prototype follows a simple client-cloud-client architecture with real-time synchronization. Conceptually, the architecture can be represented by two smartphones (Sharer and Tracker) on the sides, a Firebase real-time Database in the middle, and Google Maps services attached to the follower device.

Sharer device (mobile browser)

- Runs `sharer.html` and `sharer.js`.
- Uses the HTML5 Geolocation API (`watchPosition`) to capture GPS coordinates every few seconds.
- Sends a JSON object {`lat`,`lng`,`speed`,`timestamp`,`accuracy`} to the path `liveLocations/{sessionId}` in the firebase real-time database using the Web SDK

Firebase Real-time Database

- Acts as the central hub for location data.
- Stores one node per active session at `liveLocations/session_XXXX`.
- Pushes updates instantly to all connected clients using real-time listeners (implemented internally using WebSockets/long polling).

Tracker (Follower) device (mobile browser)

- Runs `index.html` and `tracker.js`.
- Uses HTML5 Geolocation to track its own position continuously.
- Subscribes to `liveLocations/{sessionId}`: on each database change it updates the red "sharer" marker, and on each of its own GPS updates it moves the blue "tracker" marker and keeps the camera centered on the follower.

Google Maps JavaScript API and Directions API

- Embedded only in the follower page.
- Renders the base map tiles, custom markers and navigation-style camera.
- On each position update (from either sharer or follower), the `DirectionsService.route()` method is invoked to re-compute:
  - the blue polyline path between the current tracker and sharer positions;
  - the remaining distance and estimated time of arrival (ETA), using live traffic information when available.

3 SYSTEM COMPONENTS AND DATA FLOW

This section explains in detail how each component of the prototype works and how data flows from the sharer device to the tracker device through the cloud backend. The overall behaviour can be described as a continuous loop: the sharer captures GPS coordinates, the cloud synchronizes these coordinates in real time, and the follower renders the updated positions and navigation information on Google Maps.

3.1 Sharer Device

The sharer device is a standard smartphone running a modern mobile browser that loads the `sharer.html` page and executes the associated `sharer.js` script.

3.1.1 Session Creation

When the page loads, the JavaScript logic generates a unique session identifier of the form `session_{randomString}` and displays it to the user in a read-only input field. This identifier is shared with the tracker and is used to bind both devices to the same node in the Firebase real-time database. Each active session therefore corresponds to exactly one path `liveLocations/{sessionId}` in the database.

3.1.2 Geolocation and Permission Handling

The sharer page requests permission to access the device location using the HTML5 Geolocation API. After the user grants permission, the function `navigator.geolocation.watchPosition()` starts a continuous location stream. For each GPS fix, the browser provides a set of values including latitude, longitude, accuracy, speed, heading and a timestamp. If the user denies permission or if the sensor fails, a clear error message is shown and no data is written to the backend.

3.1.3 Publishing Location to Firebase

For every successful location update, the sharer script constructs a JSON object

```
{ lat, lng, accuracy, speed, heading, timestamp }
```

and writes it to the appropriate session node using `database.ref('liveLocations/' + sessionId).set(locationData)`. The node is overwritten on each update rather than extended, so the database always stores the most recent position of the sharer only. After each write, the user interface is updated with the current coordinates and the last update time, confirming that the sharing process is active and functioning.

3.2 Firebase Real-time Database

The Firebase real-time database acts as the cloud-based synchronization hub between all connected clients. It provides low-latency data propagation using persistent connections and event-driven listeners.

3.2.1 Data Model

The database schema for this prototype is intentionally simple. Each active session is represented by one node:

$$\texttt{liveLocations/session\_XXXX} \rightarrow \{\texttt{lat}, \texttt{lng}, \texttt{accuracy}, \texttt{speed}, \texttt{timestamp}\}.$$

This flat structure minimizes read and write costs, fits well within the free tier for experimental usage, and simplifies client logic since all relevant information for a session is contained in a single object.

3.2.2 Real-Time Listeners

Firebase maintains a long-lived connection (typically via WebSockets) between the client SDK and the backend. When the follower attaches a listener using `ref.on('value', callback)`, the backend will immediately send the current value of the node and then push every subsequent change as an event. No explicit polling is required; instead, the tracker receives location updates as soon as they are written by the sharer. If multiple followers subscribe to the same session, they all receive identical real-time updates.

3.2.3 Security Rules

During prototyping, permissive security rules (`.read: true, .write: true`) are used to simplify experimentation. In a production setting, rules would restrict read and write access so that only authenticated users or explicitly authorized followers can access a particular session, thus protecting location privacy.

3.3 Tracker Device

The follower device is another smartphone (or any browser-capable device) that loads `index.html` and executes `tracker.js`. It has two main responsibilities: tracking its own movement and rendering the sharer's movement, while continuously computing the route, distance and estimated time of arrival.

3.3.1 Binding to a Session

The tracker (follower) user pastes the session identifier received from the sharer into an input field and presses the "Track" button. The script then subscribes to the database path `liveLocations/{sessionId}` by calling `database.ref(...).on('value', callback)`. From this moment onwards, any change at that location node will trigger the callback.

3.3.2 Receiving Sharer Updates

Whenever a new snapshot arrives from Firebase, the tracker reads the stored object and updates the logical variable `sharerPosition`. The red marker on the map is then moved to this new position. To achieve smooth, Uber like motion instead of abrupt jumps, the prototype uses an interpolation function that animates the marker from its current coordinates to the new coordinates over a short time interval. The same snapshot also provides the sharer's speed and timestamp, which are displayed in the interface to give context about motion and recency.

3.3.3 Tracking Follower Movement

Independently of Firebase, the follower device also uses the HTML5 Geolocation API to track its own position via another call to `navigator.geolocation.watchPosition()`. Each new GPS reading updates the `trackerPosition` variable and moves the blue marker accordingly. The map camera is re-centered on this marker using `map.panTo(trackerPosition)`, which creates a navigation-style view where the follower remains in the centre of the screen.

3.3.4 Route, Distance and ETA Computation

Whenever either the sharer position or follower position changes, the function `calculateRoute()` is invoked. This function constructs a request for the Google Maps Directions API with the current tracker position as the origin, the current sharer position as the destination, and the travel mode set to driving. The request also includes driving options such as departure time and a traffic model to obtain traffic-aware estimates where available. The Directions service returns a route consisting of one or more legs. The prototype passes this result to a `DirectionsRenderer` instance, which draws the familiar blue polyline path between the two points on the map. Distance and estimated time of arrival are extracted from the first leg and displayed in the user interface. As either party moves, these values are recomputed, giving the tracker a continuously updated sense of how far away the sharer is and how long it will take to reach them.

3.4 End-to-End Flow

Putting the components together, the full operational flow is as follows:

(i) The sharer opens `sharer.html`, grants location permission, and receives a generated session ID.

(ii) The sharer moves; the browser streams GPS readings to the node `liveLocations/{sessionId}` in the Firebase Realtime Database every few seconds.

(iii) The follower opens `index.html`, enters the same session ID, and attaches a real-time listener to that node.

(iv) Firebase pushes each new location update to the follower as soon as it is written, while the follower also tracks its own GPS position locally.

(v) On the follower device, the sharer and tracker markers are updated, the route between them is recomputed, and the distance and ETA fields are refreshed.

This loop continues for the duration of the session, resulting in a browser-based system that closely approximates live driver tracking in commercial ride-hailing applications, but implemented entirely with web technologies and free-tier cloud services.

4 DISCUSSION

The core issue observed in current "share live location → Get directions" workflows is that the target map application receives a *static* coordinate rather than a dynamic reference to the sharer's changing position. In applications such as WhatsApp, the deep link generated for Google Maps typically encodes the latitude and longitude that were current at the moment the user tapped the location (for example, a URL that ultimately resolves to a fixed `(lat, lng)` pair). When the user selects "Get directions," Google Maps interprets this as a normal point destination and computes a route only once; subsequent movement of the sharer does not alter the URL, so the destination remains frozen, and the follower must manually reopen the live location and request directions again. This design is simple and backwards compatible with any mapping application that accepts coordinates, but it inherently cannot support continuous movement without repeated manual refreshes.

One family of solutions is to change the *semantics of the URL* so that a shared link no longer represents a single geographic coordinate, but rather a reference to a live location stream. Instead of encoding raw latitude and longitude, the URL could embed a session token, for example `https://maps.app/live?sessionId=XYZ`, that maps to a server-side record where the sharer's current position is continuously updated. When a user taps such a link, the maps client would resolve the session identifier, subscribe to a live data source, and automatically recompute the route and estimated time of arrival (ETA) as new coordinates arrive. Architecturally, this is analogous to the way our prototype stores data under `liveLocations/{sessionId}` in Firebase Realtime Database and allows multiple followers to attach listeners to the same path. This approach, however, requires coordination between the messaging application and the map provider so that both understand and honour a common "live session" URL format.

A second solution is the one implemented in this work: moving the live-location logic into an independent web application that any two users can access from a browser. Instead of relying on WhatsApp or the native Google Maps application to treat shared locations as dynamic, the prototype introduces an explicit session layer on top of standard web technologies. A sharer starts a session and streams GPS coordinates to a cloud backend using Firebase Realtime Database, while a follower joins the same session and uses the Google Maps JavaScript API together with the Directions API to render both positions, maintain a navigation-style view centred on the follower, and repeatedly compute distance and ETA as either party moves. This design demonstrates that full, bidirectional live tracking does not require modifications to the underlying Maps platform; it only needs a persistent data channel and a client capable of reacting to real-time updates.

These two directions are complementary rather than mutually exclusive. A production-grade solution could allow messaging applications to generate "live session" URLs while delegating visualization and routing to a web or native client based on the same principles as our prototype. In this sense, the prototype serves as a proof-of-concept showing how live sessions, cloud synchronization and map rendering can be orchestrated to provide safer and more intuitive following behaviour than the current static snapshot model, reducing the need for drivers to repeatedly interact with their phones while in motion.

5 CONCLUSION

This project has demonstrated that it is technically feasible to provide true live person-to-person following on top of Google Maps using only web technologies and free-tier cloud services. By combining the HTML5 Geolocation API, Firebase Realtime Database,

and the Google Maps JavaScript and Directions APIs, the prototype converts a shared location into a dynamic, continuously updating destination, allowing both sharer and follower markers to move in real time while the route, distance and estimated time of arrival (ETA) are automatically recomputed as they travel. The system addresses key usability and safety limitations of current static "share live location $\rightarrow$ get directions" workflows and illustrates how a session-based design can reduce the need for drivers to repeatedly interact with their phones while driving.

The intention of this work is not to replace existing navigation products, but to propose a concrete feature enhancement for Google Maps: a native "follow live location" mode that can be triggered from shared links or within the app itself, and that continuously routes to a moving person rather than a fixed coordinate. The implemented prototype serves as a working proof-of-concept and a foundation for further development. With access to Google's internal APIs, design guidelines and engineering support, this approach could be refined, hardened and integrated at scale, bringing safer and more intuitive live-following capabilities directly to millions of Google Maps users worldwide.