International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

TIMENEY: Save Your Money, Save Your Time – A Mobile Application for Ride Comparison Using Route-Based Fare Estimation

Srinivasa Pranay

Affiliation: BEL First Grade College, Bengaluru

Corresponding author email: srinivasapranay22@gmail.com

Abstract

Many people in India check more than one ride app before booking. They type the same pickup and destination again and again and compare prices by hand, which wastes time. This paper presents TIMENEY, a mobile app built with React Native and Expo. The user enters a trip once and sees estimated ride options from Uber, Ola and Rapido on one screen. The app takes the user's position from the phone's GPS and gets the road distance and expected travel time from the Google Routes API. It then works out an estimated fare for each provider using a simple formula made of a base fare, a charge per kilometre, a charge per minute, an extra fee and a minimum fare. The cheapest option is highlighted. When the user picks a provider, the app opens that provider's app or website. TIMENEY does not book rides or take payments, and it does not scrape provider apps or use private APIs. Because of this, every fare is shown as an estimate and not as a live price, and travel time is kept separate from the time a driver takes to arrive. The paper explains the app design, the fare formula and the handoff to each provider, gives a worked example, and lists limits such as fare values that are not yet tuned and the lack of surge pricing. The work shows that a simple, legal comparison tool can save riders effort and can later use official provider APIs.

Keywords: Ride-Hailing, Fare Estimation, Mobile Application, React Native, Route Planning.

I. INTRODUCTION

Ride apps are now a normal part of daily travel in Indian cities. Uber, Ola and Rapido offer cars, auto-rickshaws and bike taxis, and the price for the same trip can be different on each app. A rider who wants the cheapest fare ends up opening one app after another, typing the same pickup and destination each time, and trying to remember which price was lower.

This is slow and boring, and it is worse when the rider is in a hurry. The problem is not that there are too few choices. The problem is the effort it takes to compare them. TIMENEY, short for "Save Your Money, Save Your Time", tries to fix this. The user enters the trip once and sees all the options side by side.

TIMENEY is only a comparison and handoff tool. It does not book rides, take payments or show live data from the providers. Most of the provider data that a comparison needs is not open to the public, and scraping or reverse-engineering the apps could cause legal problems and would break easily. So the first version of the app (the minimum viable product, or MVP) uses real route data from a mapping service and a simple fare formula that can be changed. Every price is marked as an estimate.

This paper makes four contributions. First, it describes the design of a legal ride-comparison app that uses route data and adjustable fare settings. Second, it explains the fare formula and the settings for each provider. Third, it shows how the app hands the user over to each provider, using official links where they exist and a safe fallback where they do not. Fourth, it discusses the limits of showing estimated fares and what to be careful about.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

II. LITERATURE REVIEW

Earlier studies have looked at why people choose ride apps and how the apps set their prices. Rayle et al. [1] compared ride apps with taxis and public transport in San Francisco and found that things like waiting time and convenience affect what riders choose. Cramer and Krueger [2] looked at how well ride-app drivers use their working time compared with taxi drivers. These studies show that riders care about more than the price alone, so it makes sense to show more than one detail for each option.

Prices on these apps also change with demand. Because of this, a fixed formula cannot match a live quote exactly. TIMENEY therefore shows its results as estimates and does not claim to match a provider's price at any moment.

Comparison tools help buyers in general. When it is easy to check several prices, people can find a better deal without much effort. TIMENEY applies the same idea to booking a ride.

On the technical side, the app needs the real road distance and travel time, not a guess. The Google Routes API [3] gives the distance and the travel time for a route, including current traffic, so TIMENEY uses it instead of dividing the distance by an assumed speed. For handing over the trip, Uber explains how its links can open the Uber app with the pickup and drop-off already set [4]. Ola's developer pages describe partner services, including a ride estimate service [5]. Uber also says that such a link only passes the trip details and cannot be used to get a price or time estimate [4]. So a comparison app needs its own way to work out fares.

In the sources reviewed for this paper, there was little work on small, app-based comparison tools that follow the providers' rules. This paper tries to fill that gap with a working MVP.

III. SYSTEM DESIGN AND METHODOLOGY

A. Overview

TIMENEY works as a pipeline of six parts (Fig. 1). The user opens the app, allows location access and taps the GPS button to fill the pickup with their current position. After typing a destination, the app gets the road distance and travel time, works out an estimated fare for each provider, ranks the options and shows them. When the user picks a provider, the app passes the trip on to that provider's app or website.

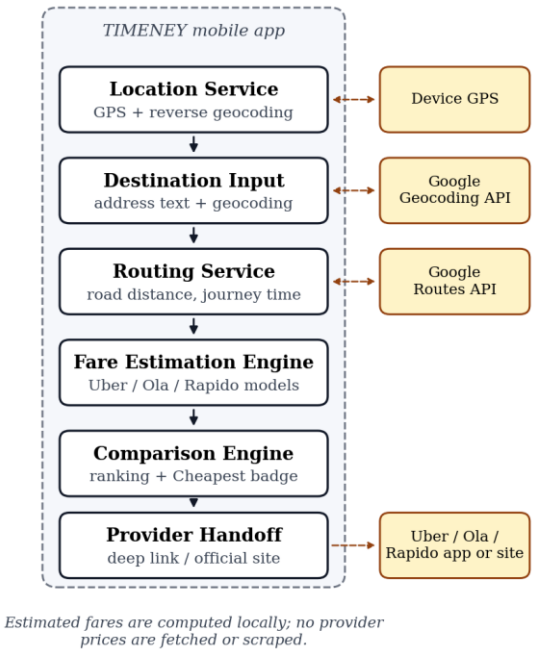


Fig. 1. Architecture of TIMENEY.

B. Location Service

The app uses Expo Location to ask for permission to use the phone's location while the app is open. If the user agrees, the app reads the phone's position and turns the coordinates into an address (reverse geocoding), so the pickup box shows a readable address instead of numbers. If the user says no, the app shows a message and the user can type a pickup address instead. That address is turned into coordinates (forward geocoding).

C. Routing Service

The distance and time come from the *computeRoutes* method of the Google Routes API. The request is for driving and uses the traffic-aware option. It asks only for the distance in metres and the duration, which keeps the reply small. The distance is converted to kilometres (d) and the duration to whole minutes (t), with a minimum of one minute. TIMENEY calls t the **estimated journey time**. It is the expected time to drive the route. It is not the time a driver will take to reach

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

the pickup point, so the app avoids any wording that sounds like a driver arrival time.

D. Fare Estimation Engine

The estimated fare for provider *p* is worked out as

$$F_p = \max(M_p, \text{round}(B_p + r_d \cdot d + r_t \cdot t + A_p)) \quad (1)$$

where B is the base fare, r_d the per-kilometre rate, r_t the per-minute rate, A the additional (booking) fee, M the minimum fare, d the road distance in km and t the journey time in minutes (every setting belongs to provider p). All amounts are in Indian rupees. All the settings are kept in one configuration file, so they can be changed without touching the rest of the code. The minimum fare copies the lowest charge that apps usually apply to short trips. The values used in the MVP are assumptions for development and are not official tariffs.

E. Comparison Engine

The estimates are saved along with each provider's name and vehicle type. The option with the lowest estimated fare gets a Cheapest badge. All providers use the same route, so the journey time is the same for every option. A Fastest badge cannot tell them apart until timing data for each provider is available. A Best Value ranking that mixes fare and time is planned but is left for the next version.

F. Provider Handoff

For Uber, the app builds a link with the pickup and destination coordinates and addresses, following the style in Uber's deep-link guide [4]. For Ola, it builds a universal link with the same trip details. For Rapido, no official link for filling in a trip was available to the project, so TIMENEY just opens Rapido's official website and does not make up any link of its own. If a link cannot be opened, the user sees an alert. The handoff does not pass on any fare. Each provider shows its own price, which may be different from the estimate.

IV. IMPLEMENTATION

The app is written in TypeScript using React Native 0.81 and Expo SDK 54. This made it possible to build it quickly and to test it on an Android phone through Expo Go without a full native build. Expo Location handles the permission and the GPS, and Expo Linking opens links to other apps. The code is split into four files: the settings (fare values and API key), the API calls (geocoding and routing), the provider handoff and the main screen. React hooks keep track of the app's state, and the fares are recalculated automatically each time a new route comes in.

The interface is one scrolling screen. At the top, an input card has the pickup box, the GPS button, the destination box and the Compare Rides button. After a comparison, a summary card shows the road distance and the travel time. Below it is one card for each provider, showing the vehicle, the estimated fare and a Continue button. A note on the screen says that the fares are TIMENEY's own estimates, not live prices from the providers, and that travel time can change with traffic. If something goes wrong, such as a missing API key, refused permission, an address that cannot be found or a failed route request, a message box explains it. Table I lists the fare settings.

TABLE I. DEVELOPMENT FARE PARAMETERS (₹)

Provider	Vehicle	Base	Per km	Per min	Fee	Min.
Uber	Go	45	14	1.50	10	70
Ola	Auto	35	13	1.25	8	60
Rapido	Bike	30	10	1.00	5	50

These values are placeholders and need to be tuned later.

The Google Maps API key is read from an environment variable. Expo public variables end up inside the app, so the key must be restricted in the Google Cloud console. For a public release, it is better to keep the key on a server.

V. WORKED EXAMPLE AND DISCUSSION

To see how the formula works, take an imagined trip of 10 km with an estimated journey time of 25 min. These numbers were picked for the example and were not measured. Table II shows the result using the settings from Table I.

TABLE II. WORKED EXAMPLE (10 KM, 25 MIN, ₹)

Provider	Base	Dist.	Time	Fee	Fare
Uber	45	140	37.50	10	233
Ola	35	130	31.25	8	204
Rapido	30	100	25.00	5	160

The fare is the total, rounded. The minimum fare does not apply here.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

With these assumptions, Rapido is the cheapest at ₹160. That is ₹44 less than Ola and ₹73 less than Uber. The order comes from the lower rates and also from the type of vehicle. A bike taxi, an auto-rickshaw and a car are not the same in comfort, space or luggage room, so the cheapest option is not always the best one. This is why the app shows the vehicle type next to each fare.

The estimates were not tested against real provider prices in this study. Checking them against prices collected by hand for a few trips is an important next step.

VI. CHALLENGES AND LIMITATIONS

- Fare values are assumptions, not researched tariffs for a particular city
- Surge pricing, tolls, taxes, waiting charges and offers are not included
- Journey time is the same for every provider, so a real "fastest" ranking is not possible, and it is not the driver's pickup time
- Providers control their link formats and can change them, so the links must be tested on different phones and app versions
- The Rapido handoff only opens the official website and does not fill in the trip
- The API key sits inside the app, and route requests may cost money
- Location data is personal and needs to be handled carefully

VII. FUTURE RESEARCH DIRECTIONS

Future work can focus on:

- Tuning the fare values using real fares from specific cities
- Adding Fastest and Best Value ranking using a score that mixes fare and time
- Using partner APIs, where providers allow it, to replace estimates with real quotes
- Filters for vehicle type and auto-complete for the destination
- A backend server with caching to protect the API key and cut costs
- A usability study with riders and an accuracy check against provider prices

VIII. CONCLUSION

TIMENEY shows that the effort of opening several ride apps can be cut down with a simple comparison tool built from legitimate parts: the phone's GPS, a commercial routing service, an open fare formula and official handoff links where they exist. The app is honest about what it knows, because it labels every price as an estimate and keeps journey time separate from driver arrival time. Its main limits are fare values that are not yet tuned and the lack of live provider data. Tuning the values, adding better ranking and using partner APIs can fix these, and this MVP is a good base for that work.

REFERENCES

[1] L. Rayle, D. Dai, N. Chan, R. Cervero, and S. Shaheen, "Just a better taxi? A survey-based comparison of taxis, transit, and ridesourcing services in San Francisco," Transport Policy, vol. 45, pp. 168–178, 2016.

[2] J. Cramer and A. B. Krueger, "Disruptive change in the taxi business: The case of Uber," American Economic Review, vol. 106, no. 5, pp. 177–182, 2016.

[3] Google, "Routes API overview," Google Maps Platform Documentation. [Online]. Available: https://developers.google.com/maps/documentation/routes (accessed Oct. 2026).

[4] Uber Technologies, "Introduction to deep links," Uber Developers. [Online]. Available: https://developer.uber.com/docs/riders/ride-requests/tutorials/deep-links/introduction (accessed Oct. 2026).

[5] ANI Technologies (Ola), "Ride estimate," Ola Developers. [Online]. Available: https://developers.olacabs.com/docs/ride-estimate (accessed Oct. 2026).

(This work is licensed under a Creative Commons Attribution 4.0 International License.)