A source-bounded transit interface for reducing cognitive load in local bus use

Alan N. Pham[1]

[1]AO Labs, Worcester, Massachusetts, USA

Live bus trackers expose route, vehicle, stop, and prediction data, but a rider may still need to translate those data into a personal action. We built a location-first WRTA interface that binds saved Worcester destinations to nearby stops, low-walk direct or one-transfer bus options, route geometry, current location, and walking estimates on one stable map. The contribution is not a new transit authority data source. It is a source-bounded interaction layer that separates official stop-time predictions from derived destination-arrival estimates, suppresses impractical long-walk options, chooses the physical stop, transfer pair, and route direction that can actually carry the rider toward the selected place, advances past missed live-predicted options, reduces map interpretation work, and makes uncertainty visible for a rider who wants fewer choices and less context switching.

1 Introduction

Public transit information systems often contain the necessary facts without presenting the specific fact a rider needs in the moment. The official Worcester Regional Transit Authority (WRTA) tracker provides the live vehicle and route surface for WRTA service (*1*, *2*). For a rider deciding whether to leave, wait, board, or keep watching the map, however, the tracker still leaves a translation step: identify the relevant stop, connect that stop to the destination, decide which route direction matters, infer whether the moving vehicle is the one to catch, and convert an arrival prediction into a personal action.

The AO Labs bus tracker was built as a narrower interface for that translation. The public version at 'bus.aolabs.io' keeps a focused live-bus surface for familiar WRTA routes and six fixed Worcester-area destinations: 96 William Street, Alden Hall, Union Station, Chipotle on Park Avenue, Cold Stone on Main Street, and Blackstone Theaters. Planning is broader than the small live-bus filter: the server evaluates a warm set of nearby WRTA routes, including Route 30 and Route 825, when those routes reduce walking or create a useful transfer. It uses WRTA live and stop-time surfaces, Ride Guide route, stop, and schedule geometry, OpenStreetMap tiles, browser geolocation, Leaflet rendering, and OSRM routed walking paths (*3*–*9*). Straight-line walking fallbacks are not eligible to choose a boarding, transfer, or exit walk. The app is therefore not authoritative beyond the sources it reads. It is a personal layer that narrows those sources into one map and a short set of low-walk trip options.

The design motivation is neurodivergent fit. The intended rider has self-reported autism and ADHD and wants the interface to remove repeated interpretation work rather than ask for more manual planning. This paper therefore treats cognitive load as an engineering requirement: keep the map visible, keep the screen stable, name the stop, show the route, show the live bus, distinguish stop time from destination time, and avoid alarm-like interaction patterns. General web accessibility standards also emphasize predictable, operable, and understandable interfaces, but this paper makes a narrower claim: a personal transit layer can be better for a specific rider by reducing avoidable interpretation (*10*).

2 Results

2.1 A stable, location-first map changes the task

The WRTA tracker is route-first: routes and vehicles are the primary objects. The AO Labs interface is location-first: a selected place reframes the map around the rider's current location, the selected destination, the boarding stop chosen from that current location, any required transfer, the exit stop near the destination, route geometry, and live buses when browser GPS is available (Fig. 1). After destination selection, the selected trip becomes the first right-panel object and the destination chooser collapses to compact controls below it. The selected trip title gives the route sequence, destination-arrival time, and whether that title is based on a WRTA live prediction or only the schedule. The trip body is a chronological list of boarding, transfer, second boarding, and exit events rather than a stack of competing trip cards; each boarding row keeps the bus-at-stop prediction and scheduled bus-at-stop time in compact metadata below the action instead of a visually dominant table. A single next-trip control remains available for intentionally later departures. Two map controls keep the spatial task recoverable: one recenters on the rider's current location, and one restores the selected route framing after local inspection. Later GPS, live-bus, and selected-plan refreshes preserve manual panning or zooming, so route framing is an explicit action rather than an automatic interruption. This changes the task from inspecting the transit network to answering a personal question: which stop, route sequence, predicted stop time, and scheduled stop time matter for this trip? When live prediction makes a transfer infeasible, the app marks that option as missed and advances to the next feasible low-walk trip instead of preserving a stale scheduled transfer.

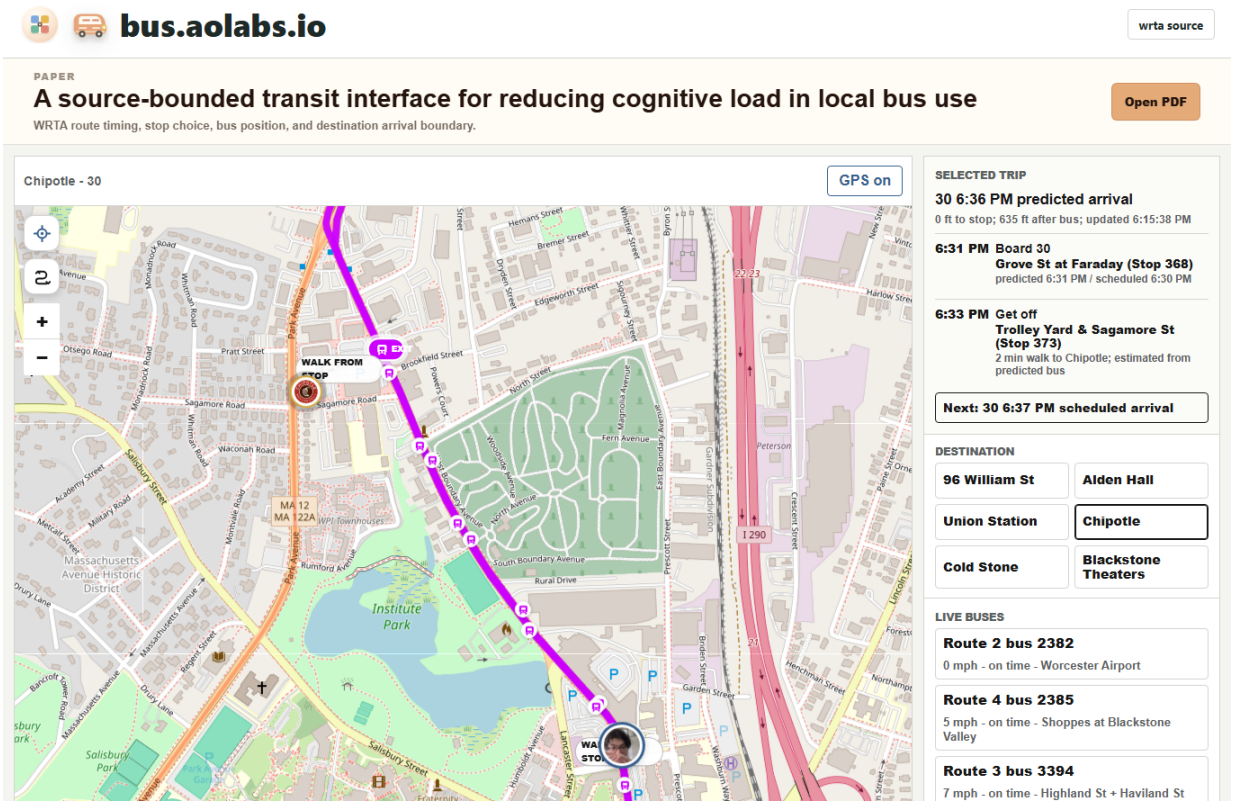

Figure 1. Stable live interface for selected WRTA destinations. The public app keeps saved destinations, route geometry, bus-stop symbols, live-bus state, current-location state, and map recovery controls on one map while the side panel keeps selected-trip state at the top of the same view. In this capture, simulated browser geolocation near WPI with Chipotle selected surfaces a Route 30 destination-arrival title labeled by prediction basis, chronological boarding and exit rows, inline predicted-versus-scheduled stop timing, and a single next-trip control when another feasible later trip exists. Compact destination controls remain below the selected trip, so the rider can change the destination without displacing the stop, route, and destination-arrival details.

Table 1. Comparison of the official source surface and the personal source-bounded layer.

Task	WRTA tracker	AO Labs bus tracker
Primary object	Route and system map	Saved place or selected destination
Stop selection	Interpreted by rider	GPS-based boarding stop, transfer stop, exit stop, or labeled fallback
Location context	Not personalized by default	Current location shown when browser GPS is available
Timing row	Stop prediction surface	Bus-at-stop time, transfer sequence, and derived destination estimate
Map behavior	General live operations view	Selected-place framing with route, transfer, and stop context
Authority	Official WRTA surface	WRTA-derived, route-bounded, not authoritative beyond sources

2.2 The app separates bus-at-stop time from destination-arrival time

The most important timing correction is semantic. A bus can arrive at a stop, and the rider can arrive at the selected destination later. If the interface collapses those events into one timestamp, the rider has to guess what the time refers to. The current app displays both values as separate claims. For the stop itself, the primary bus-at-stop time uses WRTA ‘horaire’ stop-time output, which matches the tracker popup’s predicted field, while the adjacent scheduled bus-at-stop time uses the matched ‘horaireApplicable’ row. The matcher is stop-first and direction-aware: it identifies the WRTA stop, line, and destination row for the selected physical stop and route headsign, then accepts a prediction only when the scheduled row is close enough to the selected schedule candidate to be the same bus. Candidate trips receive those matched stop predictions before the visible itinerary is ranked, so a delayed direct bus can remain available instead of being hidden behind a later transfer. The destination-arrival value is computed by adding scheduled in-vehicle travel and OSRM exit-stop-to-place walking duration, with live stop-time changes propagated when a WRTA prediction is available.

This separation prevents the app from overstating certainty. The predicted and scheduled stop-time values are transit-source values for a specific boarding stop. Destination arrival is a derived estimate for a place. It depends on walking route availability, walking speed, business entrance geometry, traffic signals, and whether the rider starts walking immediately after leaving the bus. The UI labels prediction basis, scheduled stop-time comparison, and refresh time, and the paper treats destination arrival as a derived interface estimate rather than an official WRTA prediction.

2.3 The design removes repeated interpretation work

The app is better than a general tracker for this personal use case because it removes specific repeated burdens. The rider does not need to count vague map distances, identify a stop from an unlabeled route line, infer whether a location pin is connected to a route, decide whether a long walk is being substituted for a missing transfer, or wait for a static bus marker to jump after the next feed update. The app renders route paths, bus-stop symbols, transfer context, live buses, place icons, and current location together, and it interpolates live bus markers between source updates so the vehicle appears to move continuously while still depending on WRTA data.

The interface also avoids disruptive alerts. It does not use vibration, audio, popups, emergency warnings, or alarm-like notifications. This is not a cosmetic choice. For a rider who is already tracking a bus, walking, and managing uncertainty, a calm persistent screen can be more usable than a system that interrupts attention.

2.4 Bounded superiority over the source tracker

The superiority claim is intentionally bounded. The WRTA tracker remains the official and broader source. The AO Labs app is superior only for a narrower problem: turning a small set of personal destinations into route, stop, bus, and timing context without asking the rider to interpret the full transit system. Its advantage comes from scope reduction and stable spatial framing, not from more authoritative predictions.

3 Discussion

This work shows that a transit interface can improve utility by reducing interpretation rather than adding features. For a rider with ADHD and autism, the difficult step is often not obtaining data; it is holding multiple spatial and temporal facts in working memory while the map changes. A personal tracker can reduce that cost by keeping the relevant facts on the same screen and by labeling what each time actually means.

The app also demonstrates a useful boundary for AI- or automation-adjacent personal tools: the system should not invent certainty. The correct design is not to promise exact destination arrival. It is to show the stop prediction, show the walking basis, compute a clear derived estimate, and keep the source boundary visible. This makes the app calmer and more trustworthy because the rider can see which

parts are official data and which parts are personal context.

Several limitations remain. The public app currently focuses on six saved destinations, a focused live-bus surface, and direct or one-transfer trip options rather than full-system itinerary optimization. The current default prefers trips under one mile of walking, then trips under 1.5 miles of walking, and only falls back to longer transfer options when no lower-walk trip is available in the current next-trip search window. Browser geolocation may be blocked or unavailable depending on device, browser, operating system, and permission state. WRTA stop predictions can be absent, stale, or changed by operations. OSRM walking routes may not match the exact path a rider chooses, can overstate campus shortcuts, and can fail when pedestrian network data are incomplete. When that happens, the app must reject the candidate or label the source failure rather than recommending a straight-line walk. No controlled user study has yet measured missed buses, time saved, stress, or reduction in interpretation errors. Until those observations exist, the paper presents the app as a source-bounded product and design record, not a population-level accessibility result.

4 Materials and Methods

4.1 System

The app is a Node.js web service deployed at ‘https://bus.aolabs.io/’. The browser client renders a Leaflet map with OpenStreetMap tiles. Saved destinations are fixed latitude-longitude records in the application source. The server exposes JSON endpoints for route geometry, live vehicles, trip planning, selected-location arrivals, walking routes, destination geocoding, and health state. After GPS is available, the client begins background planning requests for saved destinations so a later selection can reuse already-started or already-completed route work rather than beginning every calculation at click time. The server also warms route and common saved-trip source caches after start-up; this does not change the selected plan, but it moves WRTA and OSRM fetch latency away from the first visible destination click when the instance has just started.

4.2 Transit sources

Live vehicle state is read from the WRTA SWIV service. The app converts the SWIV ‘vitesse’ value from kilometres per hour before displaying miles per hour or interpolating marker position. Stop arrivals are read from WRTA stop-time output for the selected stop; ‘horaire’ is treated as the live predicted time shown by the WRTA tracker popup and ‘horaireApplicable’ as the scheduled time for that same row. Route stops, route shapes, and schedule details are read through Ride Guide endpoints for the WRTA dataset. The app does not create official transit predictions; it displays or derives values from those upstream sources.

4.3 Boarding-stop selection

For a selected saved destination, the primary path uses browser GPS as the trip origin. The server evaluates direct trips and one-transfer trips from WRTA route candidates, searches feasible boarding, transfer, and exit stop combinations, and rejects stop pairs whose ordered trip record does not carry the rider from boarding stop to exit stop. Transfer-pair pruning is not based on transfer walking distance alone: physically identical shared stops that are far from the rider or destination cannot crowd out a nearby transfer that actually serves the selected trip. This matters at intersections with stops on both sides of the street and on overlapping route segments: identical or similar stop names are not enough, so the chosen stop is the one whose route direction and stop sequence lead toward the selected destination or transfer. Candidate ranking balances earliest reachable destination arrival, walking distance, in-vehicle duration, transfer waiting, and a hard high-walk cutoff rather than selecting the nearest named stop by label alone; when two reachable plans arrive within a small time window, the lower-walk option is preferred. Before the final rank is exposed, the server applies live WRTA stop predictions to a bounded candidate set and recomputes feasibility, so a direct Route 31 trip at Elm Street and Russell Street is not lost simply because the static schedule made the bus appear unreachable before the prediction row arrived. The eligibility rule also preserves tight but still physically reachable trips: if leaving immediately can reach the boarding stop inside the bus grace window, the trip is shown as urgent instead of being removed as missed. This prevents a fast Route 3 plus Route 30 itinerary from disappearing merely because the ideal stand-at-stop buffer has already passed. Auto-selected options search the current service date and the next service date so late-night use shows the next available trip instead of an empty same-day state. The UI immediately frames the current location and selected destination while planning continues, then replaces that preview with the selected route, stops, stop

codes, and timing rows when the source-backed plan returns. If GPS is unavailable, the app uses a labeled location-nearest fallback instead of implying that the fallback is the rider's boarding stop.

4.4 Destination-arrival estimate

For each selected trip, the server obtains upcoming stop times for each selected boarding stop and requests OSRM walking routes from the current location to the first boarding stop, between transfer stops when needed, and from the exit stop to the selected destination. The selected plan is refreshed in the browser every three seconds so WRTA prediction changes can update the same trip panel without a manual destination click. If OSRM returns a routed path that is much longer than the direct distance, the app usually keeps the routed path and labels that basis instead of drawing a direct connector through an obstacle. A narrow exception handles very short local stop walks: if the stop is within 280 m by direct distance and the OSRM path is more than 2.7 times longer, the app uses a conservative direct-distance walking estimate and labels the over-route. This prevents a local routing artifact from hiding a bus that the official tracker and the rider can see at the nearby stop. If OSRM fails, that walk is not eligible for selected-trip planning. Destination arrival for a two-bus trip is calculated as

$$T_{\text{destination}} = T_{\text{board1}} + D_{\text{ride1}} + D_{\text{transfer}} + W_{\text{transfer}} + D_{\text{ride2}} + D_{\text{exit walk}},$$

where T_{board1} is the WRTA-derived first boarding-stop time, D_{ride1} and D_{ride2} are scheduled in-vehicle travel times adjusted by matched WRTA live stop prediction when available, D_{transfer} is the wait after the transfer walk, W_{transfer} is the transfer walking duration, and $D_{\text{exit walk}}$ is the walking duration from the exit stop to the selected place. Direct trips are the same expression with the transfer terms removed. Diagnostic walking endpoints can still return a straight-line failure geometry with 'sourceOk: false', but selected trips require a source-accepted routed walk. The UI labels destination arrival as a source-bounded derived estimate rather than an official WRTA prediction.

4.5 Visual verification

The screenshot in Figure 1 was generated from the local app at desktop width using simulated browser geolocation near WPI with Chipotle selected. This verifies that the public interaction model has the expected ingredients for a selected trip: current-location marker, selected destination, live-bus layer, selected-trip panel placed above destination controls, route sequence, route-reset control, current-location control, chronological trip rows, inline predicted-versus-scheduled stop timing, and a single

later-trip control instead of a stack of competing options.

5 Acknowledgements

The app was shaped by repeated personal corrections from Alan Pham around map framing, stop labeling, icon recognition, timing ambiguity, GPS visibility, and the need for a stable low-interruption screen.

6 Funding

No external funding supported this work.

7 Author contributions

Alan N. Pham specified the use case, evaluated the interface, and authored the design requirements. AO Labs implemented the public app, generated the manuscript artifact, and performed verification.

8 Competing interests

The author declares no competing interests.

9 Data availability

The public interface depends on WRTA live vehicle and stop-time outputs, Ride Guide route and stop geometry, OpenStreetMap tiles, browser geolocation, and OSRM walking estimates. The live surface is available at ‘https://bus.aolabs.io/’.

10 Code availability

The code is maintained in the AO Labs bus app repository. Public repository access is not currently advertised on the app page.

11 Additional information

This manuscript is a product and design record for a personal transit interface. It does not present a controlled user study, clinical result, official WRTA prediction system, or guarantee of arrival-time accuracy.

12 References

1. Worcester Regional Transit Authority, *WRTA Bus Tracker* (2026; https://swiv.wrta.cadavl.com/SWIV/WRTA).
2. Worcester Regional Transit Authority, *Worcester Regional Transit Authority* (2026; https://therta.com/).
3. Ride Guide, *Ride Guide Transit Data Interface* (2026; https://api.app.ride.guide/).
4. MobilityData, *General Transit Feed Specification: Schedule Reference* (2026; https://gtfs.org/documentation/schedule/reference/).
5. MobilityData, *GTFS Realtime Best Practices* (2026; https://gtfs.org/documentation/realtime/realtime-best-practices/).
6. OpenStreetMap contributors, *OpenStreetMap Legal FAQ and Attribution Guidance* (2026; https://www.openstreetmap.org/about/legal).
7. Leaflet, *Leaflet: a JavaScript library for interactive maps* (2026; https://leafletjs.com/).
8. World Wide Web Consortium, *Geolocation API* (2026; https://www.w3.org/TR/geolocation/).
9. Project OSRM, *OSRM API Documentation* (2026; https://project-osrm.org/docs/v26.4.0/).
10. World Wide Web Consortium, *Web Content Accessibility Guidelines 2.2* (2026; https://www.w3.org/TR/WCAG22/).