Inclusive Segment Cards (Diversity)

User Group 1: Screen Reader/Keyboard-Only Users
- **Role & Context:** Daily end-users of websites who rely on screen readers or keyboard navigation due to blindness or motor impairments. They often encounter inaccessible UI patterns like *keyboard traps* (unable to tab out of an element) which can leave them stuck[1]. This group's experience with TrapAlert is indirect (they use client sites that include TrapAlert SDK).
- **Privilege & Exclusion:** Low *ability* privilege in web navigation (visual/motor). Many sites default to mouse users, so keyboard-only users face disproportionate barriers. Lack of keyboard accessibility is consistently among the top five most problematic web issues for screen reader users[2], a problem persisting for 14+ years[3]. This indicates systemic neglect of this group's needs.
- **Needs & Pains:** Need smooth, *uninterrupted navigation* via keyboard (no traps, clear focus order). Pain points include being unable to proceed or having to abandon tasks due to inaccessible controls. For example, a poorly coded form or modal can **"trap"** focus and render a site unusable without a mouse[4], causing extreme frustration[5]. This group often feels **invisible** in analytics ,their struggles don't register unless explicitly tracked.
- **Gains from TrapAlert:** TrapAlert promises to *give them a voice*. By detecting events like "Dead-End Tab" (15+ tabs with no focus change) or repeated focus loops, it flags critical accessibility failures as they happen. Developers get concrete evidence of where these users struggle, prompting fixes. Ultimately, this leads to more inclusive designs (e.g. removing a keyboard trap immediately improves compliance with WCAG 2.1.2[6]). When such barriers are removed, these users can complete tasks independently, improving their online autonomy and trust in the product.
- **User Story:** *"As a blind user navigating with a screen reader, I want websites to let me move freely through all content via keyboard, so that I never get stuck on an element and can accomplish my goals without frustration."*

User Group 2: Users with Cognitive or Neurodiverse Conditions
- **Role & Context:** End-users who may have ADHD, dyslexia, or other cognitive conditions that affect focus and patience online. They might become *frustrated or confused* quickly by unintuitive UI flows. For instance, an overly complex form or an unexpected page behavior can prompt them to rapidly click around or give up. They often won't articulate these frustrations via feedback forms ,they'll just leave[7][8].
- **Privilege & Exclusion:** They may have average physical abilities (sight, mobility) but are disadvantaged by designs that assume high cognitive load tolerance or flawless attention. Socially, their struggles are often misunderstood as "user error"[9]. This group's frustration is frequently overlooked, as they might not self-identify as having a disability but still benefit from inclusive design.
- **Needs & Pains:** Need **clarity and guidance** in workflows. They benefit from clean layouts, clear instructions, and forgiving UI (e.g. the ability to easily correct mistakes). Pains include *information overload* or "mystery meat" UI that causes confusion ,e.g. if they

click a button and nothing happens, they might rapidly reclick in doubt (a classic *rage click* pattern[10]). Abandoning an input or backtracking repeatedly ("U-turn") often indicates they lost confidence in the flow.
- **Gains from TrapAlert:** TrapAlert's **BehaviorEngine** picks up signals like rage clicks (5+ rapid clicks) or input abandonment (start typing then delete and leave) which are strong proxies for confusion or frustration. By reporting these, the product helps developers pinpoint where users struggle to understand the interface. The outcome is improved UX ,maybe simplifying a form step, adding helper text, or fixing a dead button. These improvements reduce cognitive burden and make the experience smoother for everyone.
- **User Story:** *"As an easily overwhelmed user, I want interfaces that guide me and don't mislead or confuse, so that I can complete my task without getting frustrated or giving up halfway."*

User Group 3: Front-End Developers / Accessibility Specialists
- **Role & Context:** *Primary users* of the TrapAlert **Admin Dashboard** ,these are the people who will consume the reports and session replays. They might be web developers, QA engineers, or a11y specialists responsible for maintaining a site's usability. Their goal is to identify and fix UX problems that hinder users. They operate under time and quality pressures to improve the product continuously.
- **Privilege & Exclusion:** This group typically holds high *technical* privilege ,they have the power to change the product. However, they may lack lived experience of disabilities, creating a perspective gap. They risk *unintentional exclusion* if they design solely for themselves or "average" users. TrapAlert helps **check their privilege**: it confronts them with real evidence of where marginalized users struggle, ensuring those voices are heard in development.
- **Needs & Pains:** Need efficient, actionable insights into usability issues. Traditional analytics might show a drop-off, but not *why* users struggled. Manually testing for all edge cases (screen reader navigation, etc.) is time-consuming and sometimes overlooked. Pain points include missing an issue until it becomes a complaint or lawsuit. Also, teams often address reported issues from vocal users, while silent struggles (e.g. a keyboard-only user quietly failing) go unnoticed[11]. They need a way to catch those early.
- **Gains from TrapAlert:** TrapAlert essentially acts as their **"accessibility smoke detector."** It alerts them to friction hot-spots in real time. For example, if a user encountered a keyboard trap or a dead-click, the developer can replay the session and see exactly what went wrong. This shortens feedback loops dramatically ,instead of waiting for a user report (which 57% of frustrated users may never send[12]), the issue surfaces automatically. By using TrapAlert, teams can proactively fix problems, improve compliance, and enhance UX for all. This not only aids disabled users but also benefits overall customer satisfaction, as fixing accessibility often improves general usability (e.g. better form design helps everyone[13]). Moreover, it can protect the organization from legal risks by catching non-compliance early.
- **User Story:** *"As a front-end developer committed to accessibility, I want a real-time alert and replay when any user struggles (for example, gets stuck or rage clicks), so that I can*

quickly understand the issue and fix the underlying cause ,creating a smoother experience for all our users."

Critical Reflection Cards (Inclusion)

Using **Critical Reflection Cards**, we "stress-tested" the TrapAlert concept against challenging scenarios across individual, social, and environmental dimensions. This exercise forced us to look beyond the happy path and consider edge cases, potential harms, and unforeseen contexts for our design. Below are a few selected scenario cards and what we learned from each:

Scenario 1 ,Individual Perspective: *"Stuck and Watched"*

Card Context: *An end-user with a disability is struggling on a client website, unaware that their frustrating experience is being recorded.* In this scenario, imagine a keyboard-only user who hits a focus trap on a webpage. They press Tab 20 times, going in circles. TrapAlert detects the *Dead-End Tab* pattern and silently begins recording the DOM and user actions for a session replay. The user, however, remains stuck and eventually abandons the task in frustration.
Reflection: This scenario revealed an ethical tension: **TrapAlert currently helps the *developer*, not the user, in real-time.** The struggling user gained no immediate relief ,their session was recorded, but they still left frustrated. We discussed whether TrapAlert should intervene client-side when it detects a critical issue (for example, popping up an accessible tip like "Press Esc to exit the dialog" when a keyboard trap is detected). Such a feature could empower users directly, but it must be careful not to be intrusive or break the page. We also considered the *consent* aspect: users might not know this monitoring is happening. While it's meant to improve the site later, some users might feel uncomfortable if they knew their erratic keystrokes were being tracked. To address this, **transparency** and opt-in mechanisms could be important. The team reflected on building in a **user-facing benefit** (like optional live assistance or at least a notification like "Having trouble? We'll report this anonymously to help improve the site"). This card prompted us to think about balancing *immediate user aid* vs. *post-hoc developer insight*, and the importance of not treating users as mere data points.

Scenario 2 ,Social Perspective: *"Privacy and Trust in Session Replays"*

Card Context: *A scenario where session recording is perceived as surveillance, potentially violating privacy norms*. We considered the case of a user filling out a form that includes personal information (email, phone, etc.) and encountering frustration (say, an input error that isn't explained). TrapAlert captures a video replay and DOM snapshot at the moment the user gives up. What if that snapshot inadvertently includes the user's personal data on-screen? Moreover, we imagined the user's reaction upon learning that their keystrokes and clicks were logged and sent to a server.

Reflection: This raised **privacy and legal concerns**. Recording user sessions ,even for good intent ,edges into privacy-sensitive territory. There have been real lawsuits regarding session replay tools violating wiretapping laws when done without proper consent[14][15]. Our reflection highlighted that TrapAlert's design must be *privacy-first*. Concretely, we need to mask or avoid capturing any Personally Identifiable Information (PII) in the recordings. Perhaps the DOM snapshot should remove text inputs, or the system should log *which element* frustrated the user rather than the user's actual input. We also discussed **user consent**: at minimum, privacy policies should disclose this kind of monitoring. Ideally, TrapAlert could have settings for anonymization ,e.g. blur out text, don't record actual keystrokes, only record the fact an error occurred. On the trust side, if implemented right, TrapAlert could increase user trust in the long run (sites that improve accessibility demonstrate care). But if done opaquely, it could backfire. As a mitigation, we considered an *opt-in beta program* where users with disabilities volunteer to have their sessions tracked more closely to improve the product, thus flipping it into a collaborative angle. This card underscored the importance of **ethics in data collection**, reminding us that inclusion also means respecting user agency and privacy.

Scenario 3 ,Environmental & Systemic Perspective: *"Data Load at Scale"*

Card Context: *Our product is deployed on a large platform with millions of users ,what is the environmental footprint and performance impact?* TrapAlert relies on sending behavioral data (including videos of user sessions) to the server. We imagined a scenario where a popular e-commerce site installs TrapAlert on all pages. During a big sale, thousands of users are furiously clicking (some encountering frustration signals). This leads to a surge of data streaming to TrapAlert Server ,many session replays, snapshots, and events. On the client side, users on slower devices or networks might also feel the impact.

Reflection: We realized that **scalability and sustainability** must be considered in our design. Capturing rich data at high frequency can tax both the network and user devices. For example, Pendo's session replay guide notes that enabling replay increases data transmission from every 2 minutes to as often as every 5 seconds[16][17]. That kind of continuous upload can affect page load times and even user battery life on mobile[18]. Multiplied across millions of sessions, the cloud storage and processing has a **carbon footprint** too. Our team discussed solutions: implementing *sampling* or *rate-limiting* (not every frustrated click needs a full video, perhaps). We can also compress data and offload processing to background threads to minimize user-side performance hits. Environmentally, storing only what's necessary (e.g., last 30 seconds around a frustration event rather than a full session) can cut down on waste. We considered an adaptive approach: if the system detects too much load, it could temporarily raise the threshold for recording, focusing on the worst incidents. This scenario also brought up the idea of **value trade-off** ,yes, additional data use has an environmental cost, but the value is making software more efficient and accessible. In the long run, better UX could mean users complete tasks faster (less energy spent per task) and fewer redundant server calls (e.g., fewer error submissions, support tickets). Nonetheless, this card reminded us to build

TrapAlert as **lightweight** as possible and perhaps offer clients a “green mode” configuration (lower data, just send textual logs of events) if they prefer a smaller footprint.

Other reflection cards explored situations like **“malicious use”** (what if a developer misuses TrapAlert data to blame or shame users instead of fixing issues? ,leading us to reinforce data being used constructively and with empathy) and **“extreme contexts”** (such as use in high-stakes environments like a healthcare app ,highlighting the importance of reliability and not adding any instability). Across all these, the critical reflections spurred **innovative fixes** (e.g., privacy masking, real-time user hints) and ensured we address not just functional requirements but also ethical, social, and environmental responsibilities in our design.

Systemic Journey Map (Sustainability)

The **Systemic Journey Map** for TrapAlert extends a basic user journey by incorporating the wider social and environmental impacts at each stage. We mapped out the end-to-end experience of how TrapAlert integrates into a digital product’s lifecycle ,from a user encountering a barrier to the long-term systemic changes if such a tool is widely adopted. Below we outline the key stages of this journey, including *first-order effects* (immediate outcomes), *second-order effects* (downstream results), and *third-order effects* (broader systemic consequences):

Stage 1: Encountering an Accessibility Barrier
- **User Action & Goal:** A user (say, our screen reader user from Group 1) attempts to perform a task on a website, for example, filling out a signup form. Their goal is to submit their information and proceed.
- **Pain Point:** They hit a barrier: the **submit button is not activating** via keyboard, or focus gets trapped in the form modal. This is a classic first-order problem ,the user is momentarily *stuck* and frustrated. In a world *before TrapAlert*, they might just abandon here.
- **Immediate Social Impact:** The user experiences exclusion ,a feeling of *failure and frustration* that the site is not made for them. They may curse under their breath or feel that they are unwelcome, which can harm brand perception. In fact, such barriers drive users away; most will simply leave rather than report the issue (only ~43% might leave feedback[12]). The **social cost** is a loss of trust and a potential widening of the digital divide (a disabled user unable to sign up means they’re left out of that service).
- **Immediate Environmental Impact:** At this exact moment, the environmental footprint is minimal, just the energy for the user’s device as they try and retry actions. However, if they attempt many times or reload pages, that’s extra server calls and energy. If they give up and seek an alternative (like a call center or physical paperwork), that might incur more resources in the big picture. This highlights that *inefficient UX can indirectly cause more resource usage*.

Stage 2: TrapAlert Detection & Reporting
- **What Happens:** TrapAlert's SDK, running in the background of the site, recognizes the frustration signal, e.g., 5 rapid clicks on the disabled submit button (*rage clicks*) or the user pressing Tab repeatedly with no effect (*dead-end tab*). It immediately starts capturing the event data. A **session replay** video is recorded, and the DOM snapshot around the broken form is taken. This package is sent to the TrapAlert Server (via an API endpoint) along with metadata like the tenant ID and a *"struggle score"* indicating high frustration.
- **First-Order Effect (Tech):** There is a slight performance cost here. The user's browser now uploads data in the background. Thanks to conscious design, TrapAlert uses compression and a low-priority thread, but as noted earlier, frequent data transmission (especially if many events) can tax the network[17]. We've minimized impact, so the user likely doesn't notice any slowdown. The *shadow DOM UI* might also kick in, for instance, a subtle sidebar could appear to acknowledge the struggle (depending on configuration).
- **Social Impact:** Immediately, the user might not be *directly* helped (unless we implement the real-time hint feature discussed). However, unbeknownst to them, their difficulty is **not in vain**: it's being logged to *advocate for them* to the developers. From a consent perspective, hopefully the user was informed via the site's privacy notice. There's a potential *trust issue* if they were not; users generally don't like feeling monitored without consent. But if framed correctly (e.g., "We collect anonymous struggle data to improve our site"), this can be acceptable. The **positive social impact** here is that an *invisible observer* noticed a marginalized experience that would otherwise be lost. Instead of the user silently exiting, their experience will contribute to a fix. This begins to shift the culture: it treats *accessibility issues as first-class data*, just like errors or crashes, highlighting that these frustrations matter.
- **Environmental Impact:** The data sent (video, DOM snapshot) consumes bandwidth and server storage. Multiplied by many users, this is the stage with the largest environmental footprint. Our systemic map flags this as a *trade-off*: more data for better UX vs. energy use. TrapAlert mitigates this by capturing short clips and only when thresholds are exceeded (reducing volume). Still, the cloud storage and processing of many videos do have a carbon cost. At this first-order stage, it's a small addition per user, but we note it for aggregate impact later.

Stage 3: Developer Investigates via Admin Dashboard
- **Action:** On the TrapAlert Dashboard, the product team gets a ping: a new high-frustration session was recorded. A developer or QA opens the dashboard and sees the **"struggle session"** with a high score. They replay the user's session video: it clearly shows the user trying to click "Submit" repeatedly and nothing happening, then attempting to Tab and getting stuck in the form. The exact DOM element causing the trap is highlighted by TrapAlert (perhaps a certain `<div>` overlay capturing focus).
- **Second-Order Effect (Product Team):** This is where insight turns into action. The developer experiences an **"empathy moment."** Watching the replay, they *feel* the user's frustration. Instead of a vague bug report, they see it from the user's eyes. This often spurs quick action it's now very clear what went wrong and why it's a bad experience. According to industry insight, reducing such frustrating experiences not only helps users but also

correlates with higher conversion and retention for the business[19]. The developer knows fixing this could save customers. Socially, within the team, this fosters a more inclusive mindset: team members start to *proactively think about accessibility* to avoid seeing more struggle sessions. It's almost like having users with disabilities as honorary team members whose experiences are present in the data. This stage can shift internal power dynamics: the needs of marginalized users are now backed by concrete evidence and priority, rather than being dismissed as edge cases.
- **Social Impact:** The immediate social impact is internal; the team improves its practices and learns. Also, if the team communicates back (for instance, reaching out to the user if identifiable or posting a changelog "we fixed an accessibility bug"), it can build goodwill. There's also a compliance aspect: catching this now may avoid a potential ADA violation or lawsuit, which is a broader social/ethical win.
- **Environmental Impact:** Investigating and fixing bugs has some resource cost (developer time, their computer usage, maybe additional testing). But these are part of normal operations. One could argue a small positive environmental effect: by catching the issue early via data, the team avoids potentially more wasteful processes (like lengthy customer support cases, or multiple failed attempts by many users). It's hard to quantify, but efficient problem resolution is generally leaner on resources than letting issues persist.

Stage 4: Solution Implementation and User Experience Improvement
- **Action:** The development team fixes the issue. For the keyboard trap, they might add proper focus handling or remove the offending script. They deploy an update to the website. Now, when the original user (or any other with a similar setup) returns, they can successfully submit the form without incident. TrapAlert's struggle score for that flow drops to zero as no frustration events occur.
- **Second-Order Effect (User Experience):** The **user's journey is improved**. What was a dead-end is now a smooth path. The originally frustrated user might never know why it's better, but they'll feel the difference, they accomplish their task, which could mean they become a customer, or get the service they needed. On a larger scale, each fix makes the product more accessible not only for the specific user group who struggled but often for everyone. For example, adding a visible focus outline or a "Skip to Content" link benefits screen reader users primarily, but sighted keyboard users and even power users enjoy the efficiency[20]. This is a classic *curb-cut effect* in digital form: solving for the minority yields convenience for the majority.
- **Social Impact:** The product has become **more inclusive**. That means more people can engage with it, which has social ripple effects: a broader user base, more diverse perspectives among users, and a reputation for accessibility. An immediate measurable impact might be reduced drop-off rates on that form, more conversions from users with assistive tech. Over time, as these changes accumulate, the company might see fewer complaints about accessibility, and perhaps even positive feedback from users who notice the site "just works" for them. Internally, success stories like this reinforce the value of *conscious service design*. It becomes part of the team's ethos to catch and fix barriers proactively.
- **Environmental Impact:** A smoother UX can have minor positive environmental effects. If

tasks take less time and fewer retries, that's fewer server requests and less energy per transaction. For instance, if 1,000 users previously attempted the form twice on average (2,000 submissions, with one failed attempt each), and now it's 1,000 successful single submissions, that's roughly half the processing and data transfer for the same outcome. Multiply that efficiency by many fixes and many users, and it's not insignificant. Also, when digital services are accessible, more people can use them *instead of* resource-intensive alternatives. A user who succeeds in an online form doesn't need to drive to an office or make a paper request, saving travel and materials. These are small per instance, but they add up as inclusion scales.

Stage 5: Systemic and Third-Order Effects

- **Wider Adoption:** Now consider TrapAlert's approach adopted across many organizations. Industry-wide, if **behavioral observability for accessibility** becomes standard, the web ecosystem could shift. Common frustration patterns (like our five heuristics) might be formally recognized as things to avoid. Perhaps WCAG guidelines in the future include advisory techniques for preventing "rage click" inducing designs or focus traps, informed by data from tools like TrapAlert.
- **Third-Order Social Impact:** At a societal level, such tools can contribute to a culture where **digital inclusion** is simply part of quality. Users with disabilities might start expecting that if something is frustrating, it will be noticed and fixed, rather than feeling they're shouting into the void. It could empower advocacy: organizations representing users with disabilities could use aggregated TrapAlert data to argue for better practices ("Look, across 100 sites, screen reader users consistently got stuck on carousels… fix this!"). In an ideal outcome, the web becomes a more **equitable space**, closing the gap in access. More people can participate in e-commerce, e-learning, e-government without facing silent struggles. This aligns with the principle that *inclusive design benefits everyone*. As the A11Y Collective noted, designing with keyboard users in mind creates better experiences for all users, not just those with disabilities[21][1]. Over time, we might see higher employment and education rates among people with disabilities as digital barriers fall.
- **Third-Order Environmental Impact:** Systemically, the impact here is nuanced. On one hand, a fully digital-inclusive society means more people using digital services (which has an energy cost). But it also means we maximize the utility of existing infrastructure by serving *all* users, possibly reducing wasteful parallel systems. For example, if a government service is accessible online, fewer people need to drive to an office (saving transportation emissions). Inclusive tech can thus support sustainability by optimizing resource use. Additionally, by monitoring and improving UX continuously, companies may iterate towards *leaner* interfaces; perhaps less bloat, which can also reduce data payloads. The third-order effect might also involve **tech lifecycle**: if frustration is reduced, devices and software might have longer user loyalty and life (people don't abandon or constantly switch apps out of frustration, which can indirectly reduce electronic waste). These links are speculative but worth noting: sustainability is not just ecology, but also maintaining systems that serve humans effectively so that extra processes aren't needed.

Data & Insights: Throughout this journey mapping, we backed our claims with data where possible (e.g., known frustration signals, survey results about persistent issues) to validate each step. For instance, knowing that *lack of keyboard access* is a longstanding pain point[2] helped justify Stage 1's focus, and understanding how **frustration correlates with business metrics**[19] reinforced the importance of Stage 3 and 4 outcomes. We will continue to gather metrics as TrapAlert is used in the field, such as reduction in form abandonment rates, or the number of issues fixed.

Trade-offs and Priorities: The journey map illuminated trade-offs at each stage (immediate help vs. data collection, privacy vs. insight, thoroughness vs. performance). By visualizing first/second/third order effects, we prioritized actions that have positive downstream impact. For example, a small first-order cost (capturing an extra 100KB of data) is worth it for a second-order gain (fixing a barrier that hundreds of users would otherwise hit repeatedly). Conversely, we avoided choices that solve a short-term issue at the expense of a larger problem (e.g., we wouldn't "fix" a user's frustration by just hiding the symptom because that trades a momentary relief for a worse overall experience).

In summary, the Systemic Journey Map helped us ensure that TrapAlert's design decisions are **conscious and holistic**. It's not just about catching a single bug; it's about creating a virtuous cycle where **user struggles lead to insights, insights lead to fixes, fixes lead to inclusion, and inclusion benefits both society and the sustainability of the service**. Each phase of the journey is an opportunity to reinforce our product's core goal: *making the digital world more accessible, one frustration at a time*, while being mindful of the broader ecosystem in which this change takes place.

---

[1] [4] [13] [21] How Keyboard Traps Impact Web Accessibility - The A11Y Collective

https://www.a11y-collective.com/blog/keyboard-trap/

[2] [3] WebAIM: Screen Reader User Survey #10 Results

https://webaim.org/projects/screenreadersurvey10/

[5] [6] [20] Why Keyboard Traps Are One of the Most Frustrating Accessibility Issues

https://www.boia.org/blog/why-keyboard-traps-are-one-of-the-most-frustrating-accessibility-issues

[7] [8] [9] [10] [11] [12] [19] The Complete Guide to Fixing Customer Frustration Online | Fullstory

https://www.fullstory.com/resources/guide-to-understanding-user-frustration-online/

[14] [15] Session Replay Software: Privacy Risks and User Consent

https://verasafe.com/blog/session-replay-software-and-privacy/

[16] [17] [18] Replay performance impacts ,Pendo Help Center

https://support.pendo.io/hc/en-us/articles/18443907632923-Replay-performance-impacts