**nagAI: Bug Report**

Wooyoung Jung, Jiwoo Kim, Joohyoung Jun

1. Timer circle not resetting when stopped with hand gesture (Resolved)

| | |
|---|---|
| Frequency | Once every 4-5 timer interactions (20-25% probability) |
| Environments | Wooyoung:<br>● Samsung Galaxy Book 4<br>● WSL 2 (Ubuntu 22.04 LTS) on Windows 11 Home (24H2)<br>● npm v10.9.2<br>● React v19.1.0<br>● Google Chrome v135.0.7049.116 |
| Severity (S1-S4) / Priority (S1-P4) | ● Severity: S2 — Impairs a core user interaction (visual reset of timer)<br>● Priority: P1 — Should be resolved before feature freeze |
| Reproduction Steps | 1. Run the frontend in development mode (`npm run dev`).<br>2. Set `GEMINI_CALL_ENABLED` to `true` in `frontend/src/hooks/useBehaviorDetection.ts`.<br>3. Start up the app in the browser, enable webcam, wave hand at the webcam to start the timer, and wave again to stop<br>4. Observe the timer stopping. About once every 4-5 timer interactions, the timer display resets correctly but the circle indicator around the display does not, as if a session was still in progress. |
| Debugging Process | 1. In `frontend/src/components/Timer/Timer.tsx`, I logged internal state variables such as `isRunning`, `wasPaused`, `wasPaused` to investigate if the timer was being stopped correctly.<br>2. Seeing that the state values were transitioning correctly, I then thought if there was a way to re-render the display to reflect the updated states. |
| Root Cause Analysis:<br>a. Bottlenecks<br>b. Expected impact | ● The `progress` value is a `MotionValue<number>` used to animate a CSS custom property (`--progress`) for the circular progress bar. |

| | |
|---|---|
| | ● While `progress.set(0)` is called on stop, Framer Motion sometimes retains the old animated state if an ongoing animation isn't fully stopped before setting a new value.<br>● Since `animate(progress, 100)` runs whenever `isRunning` is toggled, it may re-trigger prematurely if `stop()` doesn't finish flushing the current animation state before the next render.<br><br>Bottlenecks:<br>a. `controlsRef.current.stop()` only halts the animation but does not always guarantee full reset of MotionValue propagation, especially under async webcam-triggered gesture updates.<br><br>Expected Impact:<br>● The circle progress indicator (which wraps around the timer digits) fails to reflect that a session has ended.<br>● This causes UI inconsistency and confusion, especially since the time display does reset to `00:30`.<br>● The user may believe the timer is still active or frozen. |
| Proposed Solutions | Flush animation and delay value reset:<br>● Ensure the current animation is fully stopped, then call `progress.set(0)` in a `requestAnimationFrame` or slight timeout to avoid overlap. |
| Follow-up Questions | ● Should we debounce the gesture input to reduce animation collisions?<br>● Would switching to `animateMotionValue(progress, ...)` with a unique animation key each time help reset correctly?<br>● Can we validate via `progress.get()` that value is being set but not reflected visually? |

2. Pause behaviorDetection while DistractionModal is displayed (Resolved)

| | |
|---|---|
| Frequency | Always; the current logic to stop behaviorDetection while DistractionModal is displayed does not work. |
| Environments | Wooyoung:<br>● Samsung Galaxy Book 4<br>● WSL 2 (Ubuntu 22.04 LTS) on Windows 11 Home |

| | |
|---|---|
| | (24H2)<br>• npm v10.9.2<br>• React v19.1.0<br>• Google Chrome v135.0.7049.116 |
| Severity (S1-S4) / Priority (S1-P4) | • Severity: S2 —Undermines modal behavior and can cause unexpected app behavior<br>• Priority: P1 — Immediate fix required to uphold UI/UX correctness and prevent side effects |
| Reproduction Steps | 1. Run the frontend in development mode (`npm run dev`).<br>2. Set `GEMINI_CALL_ENABLED` to `true` in `frontend/src/hooks/useBehaviorDetection.ts`.<br>3. Start up the app in the browser, enable webcam, wave hand at the webcam to start the timer.<br>4. Look at your phone, with the phone fully captured in the webcam. Or, any behavior signaling a distraction works.<br>5. If the Gemini API correctly detects the distraction, the DistractionModal should pop up.<br>6. Perform hand gestures to resume/stop the timer.<br>  **a. Bug:** The timer responds to the gestures.<br>  b. While we intend to prevent any interaction with the timer until the user recognizes their distraction b, the current logic does not prevent that. |
| Debugging Process | • Confirmed that `stopBehaviorDetection()` is called in `handleDistraction()`, and verified log output: ("behavior detection stopped").<br>• However, Gemini prompts and API calls **continue executing** after modal is shown, as seen in console logs of internal state variables such as `isRunning`.<br>• Verified that `isActiveRef.current` is still `true` or reset incorrectly in async loops (e.g., `loop`, `captureSnapshotAndAnalyze`) after modal is triggered. |
| Root Cause Analysis:<br>c. Bottlenecks<br>d. Expected impact | `stopBehaviorDetection()` sets `isActiveRef.current = false`, but:<br>1. **Asynchronous race condition** in `captureSnapshotAndAnalyze`:<br>  ○ If motion is detected right before |

| | |
|---|---|
| | `stopBehaviorDetection()` is called, the async Gemini analysis proceeds anyway.<br>○ By the time `isActiveRef.current` is checked, snapshot + API requests may have already been dispatched.<br>2. **No explicit check for modal visibility** in `handleBehaviorResult()`.<br>○ Even after stopping detection, responses already in-flight are processed and trigger timer actions (e.g., `pause`, `stop`, `resume`).<br><br>Bottlenecks:<br>● Lack of a reliable **global modal-active flag** checked inside `handleBehaviorResult()` and `detectMotion()`.<br>● Inability to **cancel in-flight Gemini API requests** once behavior detection is stopped.<br><br>Expected Impact:<br>● Users can inadvertently interrupt or resume timer while modal is active via gestures.<br>● Undermines the modal's purpose of freezing input and behavior.<br>● May lead to inconsistent app states or UI confusion. |
| Proposed Solutions | ● Add a `isModalVisibleRef:`<br>○ Create a new `useRef<boolean>` to track DistractionModal visibility across `useBehaviorDetection.ts` and block detection when true<br>● Hard-check `isModalVisibleRef.current` in `detectMotion()` and `handleBehaviorResult():`<br>○ Prevent all motion capture and Gemini API dispatches when modal is active<br>● Add early `return` in `captureSnapshotAndAnalyze()` and `callGeminiAPI()` if modal is visible<br>● Consider aborting Gemini API fetch requests:<br>○ Use `AbortController` to cancel any outstanding requests if modal interrupts detection<br><br>Example in `handleBehaviorResult()`: |

| | |
|---|---|
| | `if (isModalVisibleRef.current) {`<br>`  console.log('Ignoring behavior result during modal');`<br>`  return;`<br>`}` |
| Follow-up Questions | ● Should we disable gesture detection globally via app state, or just pause at the Gemini logic layer?<br>● Should the modal overlay the webcam feed visually to reinforce that detection is paused?<br>● Do we need to throttle motion events after resume to avoid instant re-triggering? |

3. Resolve inaccurate hand gesture recognition (Resolved)

| | |
|---|---|
| Frequency | ● Random (hand gesture recognition through the Gemini API is nondeterministic).<br>● Affects both fist (START/PAUSE) and palm (STOP) detection.<br>● Recognition accuracy varies between 60%–80% depending on lighting, distance, and gesture clarity. |
| Environments | Wooyoung:<br>● Samsung Galaxy Book 4<br>● WSL 2 (Ubuntu 22.04 LTS) on Windows 11 Home (24H2)<br>● npm v10.9.2<br>● React v19.1.0<br>● Google Chrome v135.0.7049.116 |
| Severity | ● Severity: S3 — Affects usability and reliability of a primary feature (gesture-based control)<br>● Priority: P2 — Requires mitigation or fallback to prevent poor user experience |
| Reproduction Steps | 1. Set `GEMINI_CALL_ENABLED = true` in `useBehaviorDetection.ts.`<br>2. Launch the app in Chrome, enable webcam.<br>3. Attempt to control the timer using gestures:<br>  a. Raise **fist** to start or resume.<br>  b. Raise **palm** to stop or pause.<br>4. Observe that:<br>  a. Sometimes the correct action is not recognized. |

| | |
|---|---|
| | b. Sometimes the wrong gesture is inferred (e.g., palm interpreted as fist). |
| Debugging Process | ● Logged raw image snapshots and Gemini API prompts<br>○ e.g.: (`console.log('Gesture detected: PAUSE')`)<br>● Verified that snapshots are being captured and encoded correctly (`toBlob()` and base64 verified)<br>● Printed raw and parsed API responses: `console.log('Gemini raw response', rawResp);`<br>● `console.log('Parsed Gemini result', parsed);`<br>● Observed frequent ambiguities or hallucinations in Gemini's interpretation of hand gestures |
| Root Cause Analysis:<br>e. Bottlenecks<br>f. Expected impact | ● The Gemini prompt relies on visual inference of hand gestures from webcam snapshots passed as `image/png` base64.<br>● The API is not optimized for consistent gesture classification, especially under:<br>○ Dim or uneven lighting<br>○ Occluded hands or low-resolution frames<br>○ Background clutter or reflective objects.<br>● Prompt ambiguity: The `running` and `paused` prompts contain multiple conditional branches, which may reduce precision in generation:<br>○ `if palm: STOP, if fist: PAUSE, else: analyze focus`<br><br>Bottlenecks:<br>● Gemini API's visual grounding capabilities are probabilistic and not gesture-specialized.<br>● Snapshot resolution (768x768) may still lack the fidelity needed for fine-grained finger recognition.<br>● No post-processing or confidence thresholds to validate inferred action.<br><br>Expected impact:<br>● Timer may **fail to start, pause, or stop** when intended.<br>● Leads to **frustration** and **inconsistent experience**, especially when combined with distraction handling or timed sessions.<br>● Reduces trust in the reliability of AI-assisted interaction. |

| | |
|---|---|
| Proposed Solutions | ● Split gestures into isolated versions:<br>○ Use dedicated prompt per action: `detect palm only`, `detect fist only`, instead of combining both in one.<br>● Improve snapshot lighting and framing:<br>○ Add a semi-transparent overlay instructing the user to center hand in frame with adequate lighting.<br>● Log ambiguous cases:<br>○ Track false positives and incorrect responses in local debug logs (e.g., wrong `action`). |
| Follow-up Questions | ● Should we implement **confidence scoring** or **manual confirm fallback** (e.g., confirm gesture before applying)?<br>● Should we **switch to MediaPipe Hands** for gesture-only use cases, reserving Gemini for higher-level distraction classification?<br>● Should snapshots be **enhanced with preprocessing**, like contrast boosting or blur detection? |

4. Unreliable motion detection for triggering Gemini API calls (Resolved)

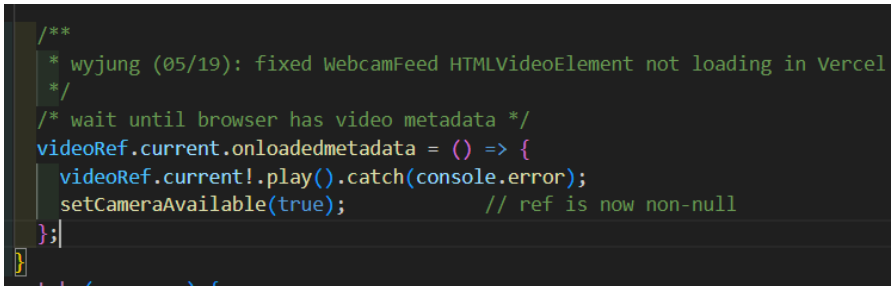
| | |
|---|---|
| Frequency | ~60% of user motion events are not registered by our pixel-diff logic, leading to missed Gemini API invocations. |
| Environments | Wooyoung:<br>● WSL 2 (Ubuntu 22.04 LTS) on Windows 11 Home (24H2)<br>● npm v10.9.2<br>● React v19.1.0<br>● Google Chrome v135.0.7049.116 |
| Severity | ● Severity: S3 — Causes degraded behavior of a critical input detection path<br>● Priority: P1 — Needs tuning before gesture detection can be considered production-stable |
| Reproduction Steps | ● Launch the frontend (`npm run dev`).<br>● Enable webcam; ensure behavior detection is enabled.<br>● Move your hand (fist or palm) rapidly within webcam view.<br>● Observe that: |

| | |
|---|---|
| | ○ Majority of motion events **do not log** as `'motion detected'`.<br>○ Gemini snapshot analysis does **not trigger** even with visible, intentional motion. |
| Debugging Process | ● Logged diff values in `detectMotion()`:<br>○ `console.log('avgDiff:', avgDiff);`<br>● Observed many motion frames yielding `avgDiff < 10`, especially when:<br>○ Background is bright or uniform<br>○ Hand motion is not close to the lens.<br>● Inspected `MOTION_DETECTION_THRESHOLD` (currently `2`) and `MOTION_SNAPSHOT_INTERVAL` (2000ms).<br>● Found that snapshots are **skipped** due to `avgDiff` not breaching hardcoded threshold. |
| Root Cause Analysis:<br>g. Bottlenecks<br>h. Expected impact | ● Pixel Difference Threshold Too Rigid:<br>○ `avgDiff > 10` is not adaptive to lighting, background, or device-specific noise.<br>○ Low-motion deltas (like gradual hand movements) are filtered out, even though they're relevant.<br>● Frame Timing Uncontrolled:<br>○ `detectMotion()` runs every 500ms (via `loop()`), but webcam frames update more frequently.<br>○ We may be comparing redundant or identical frames, reducing perceived motion.<br>● Canvas Resolution Mismatch:<br>○ Downscaling to `64xN` may blur out small motions, reducing per-pixel deltas.<br>○ Aspect ratio scaling (rounded `h`) may distort comparisons if not handled carefully. |
| Proposed Solutions | ● Lower or normalize pixel diff threshold:<br>○ Replace `avgDiff > 10` with a scaled or adaptive threshold based on historical average or noise floor.<br>● Compare frames at higher temporal resolution:<br>○ Increase `loop()` frequency (e.g., 250ms) or add logic to align with actual webcam frame rate.<br>● Add rolling window buffer for diffs:<br>○ Trigger detection if average motion exceeds |

| | |
|---|---|
| | threshold over N frames (e.g., 3 of 5). |
| Follow-up Questions | ● Should we expose `MOTION_DETECTION_THRESHOLD` and `avgDiff` threshold as dev-configurable?<br>● Would a proper background-subtraction algorithm (e.g., frame differencing with decay) be more stable?<br>● Should we run motion detection at native resolution and downsample only for transmission, not logic? |

5. WebcamFeed videoRef returns null -> webcam feed not showing on Vercel production build

| | |
|---|---|
| Frequency | Always |
| Environments | Wooyoung:<br>● WSL 2 (Ubuntu 22.04 LTS) on Windows 11 Home (24H2)<br>● npm v10.9.2<br>● React v19.1.0<br>● Google Chrome v135.0.7049.116 |
| Severity | ● Severity: S3 — Critical functionality not working<br>● Priority: P1 — Needs immediate resolution |
| Reproduction Steps | ● Deploy the frontend (`git push origin main`).<br>● If deployment does not start, call the Vercel deployment hook.<br>● Navigate to the deployed frontend.<br>● Observe that:<br>  ○ The webcam feed is not activated, and it appears as a black screen. |
| Debugging Process | ● Logged status values for videoRef, the HTML video element of the WebcamFeed component. Returned `null`.<br>● Resolved by waiting for videoRef.onLoadedMetadata() |

| | |
|---|---|
| |  |
| Root Cause Analysis:<br>i. Bottlenecks<br>j. Expected impact | • Timing issue; race condition between rendering videoRef element in the DOM and loading the metadata for the embedded video player. |
| Proposed Solutions | • Wait for videoRef.onLoadedMetadata(). |
| Follow-up Questions | • Why was this bug not observed in our local development build, but manifested in our Vercel production build? |

6. WebcamFeed component becomes tiny after pressing 'Save' in MyPage (Resolved)

| | |
|---|---|
| Frequency | Occurs every time a user navigates back to the MainPage after changing and saving their username in MyPage. |
| Environments | Joohyoung:<br>• Lenovo ideapad slim 3<br>• Windows 10 (64-bit)<br>• npm v10.9.2<br>• React v19.1.0<br>• Google Chrome v135.0.7049.117 |
| Severity (S1-S4) / Priority (P1-P4) | S1, P1 |
| Reproduction Steps | 1. Go to ‘MyPage’<br>2. Edit ‘name’ and click ‘Save’ button<br>3. Navigate back to ‘Mainpage’<br>4. Observe the webcam div - appears to be extremely small<br>5. Try refresh page - the issue does not resolve |
| Debugging Process | 1. Initially confirmed that the webcam displayed correctly before saving the user name.<br>2. Observed that navigating back to MainPage after |

| | |
|---|---|
| | saving username in MyPage caused .webcam-feed div to collapse.<br>3. Inspected that width was not properly applied.<br>4. Enforced 'width: 100% !important;' in the .webcam-feed in WebcamFeed.css.<br>5. After this change, the webcam div retained its correct size across page transitions and no longer collapsed |
| Root Cause Analysis:<br>k. Bottlenecks<br>l. Expected impact | The webcam container (.webcam-feed) relied on aspect-ratio and height: auto for layout sizing, but upon returning from MyPage, the component was re-rendered or layout context was altered, causing its width to collapse to 0 or an unintended value.<br>● Bottleneck: The '.webcam-feed' element did not retain proper width on page transition due to missing enforcement of width in CSS.<br>● Expected Impact: The webcam area collapsed completely, breaking a core feature and preventing users from using the application as intended. |
| Proposed Solutions | Explicitly set 'width: 100% !important;' in the '.webcam-feed' in WebcamFeed.css to ensure consistent layout rendering across page transitions and prevent unexpected collapse. |
| Follow-up Questions | 1. Does using '!important' for width in .webcam-feed risk overriding styles in other components or pages?<br>2. Should we add regression tests or visual checks for critical components like the webcam? |

7. Checkboxes not selectable in TermsPage (Terms and Privacy Agreements) (Solved)

| | |
|---|---|
| Frequency | Every time the TermsPage component was rendered |
| Environments | Joohyoung:<br>● Lenovo ideapad slim 3<br>● Windows 10 (64-bit)<br>● npm v10.9.2<br>● React v19.1.0<br>● Google Chrome v135.0.7049.117 |
| Severity (S1-S4) / Priority (P1-P4) | S2, P1 |
| Reproduction Steps | 1. Navigate to 'TermsPage' |

| | |
|---|---|
| | 2. Try clicking on the ‘I agree’ checkboxes under Terms and Privacy sections<br>3. The ‘Continue’ button is disabled due to ‘agreeTerms’ and ‘agreePrivacy’ remaining false |
| Debugging Process | 1. Verified that checkbox ‘onChange’ handlers were defined, but did not update state<br>2. Found that checkbox checked props were not linked correctly to ‘useState()’ values<br>3. Confirmed that in some cases, ‘agreeAll’ logic was overriding individual checkbox control<br>4. Refactored the logic to ensure checked props and ‘onChange’ handlers were properly bound to state updates |
| Root Cause Analysis:<br>m. Bottlenecks<br>n. Expected impact | Checkbox inputs had valid JSX but were not properly updating state due to incorrect handler or missing e.target.checked usage.<br>● Bottleneck: The UI appeared interactive, but user input had no effect, breaking the user flow at a critical onboarding step.<br>● Expected Impact: Users were unable to proceed to account creation, making it impossible to use the app beyond the Terms page. |
| Proposed Solutions | Rewrote the onChange logic for each checkbox to update the correct state using e.target.checked, and restructured the agreeAll logic to properly reflect combined checkbox states. |
| Follow-up Questions | 1. Do we need to test form interactivity for similar components in automated tests?<br>2. Should the agreeAll state be computed dynamically instead of stored in useState? |

8. Hover effect not showing for Cancel and Continue buttons in TermsPage (Resolved)

| | |
|---|---|
| Frequency | Every time the TermsPage was rendered |
| Environments | Joohyoung:<br>● Lenovo ideapad slim 3<br>● Windows 10 (64-bit)<br>● npm v10.9.2<br>● React v19.1.0<br>● Google Chrome v135.0.7049.117 |

| | |
|---|---|
| Severity (S1-S4) / Priority (P1-P4) | S4, P4 |
| Reproduction Steps | 1. Navigate to ‘TermsPage’<br>2. Check and agree all terms and conditions so that the ‘Continue’ button is activated<br>3. Hover over the ‘Cancel’ and ‘Continue’ buttons<br>4. Observe that no hover style is shown |
| Debugging Process | 1. Verified that the ‘.terms-cancel-button:hover’ and ‘.terms-continue-button:hover’ were defined in TermsPage.css, but no visible changes occurred on hover.<br>2. Found that inherited styles or missing specificity prevented the hover effects from being applied.<br>3. Resolved the issue by adding explicit ‘!important’ overrides for color, border-color, outline, and box-shadow in the ‘.terms-cancel-button:hover’ and ‘.terms-continue-button:hover’ styles. |
| Root Cause Analysis:<br>o. Bottlenecks<br>p. Expected impact | CSS hover styles lacked specificity or were overridden by other global styles or framework defaults, leading to no visible hover effect.<br>● Bottleneck: Users received no visual feedback on button hover, which may have caused confusion or made the UI feel unresponsive.<br>● Expected Impact: Poor UX due to lack of visual cues, especially for critical buttons like ‘Continue’ and ‘Cancel,’ potentially reducing user confidence in proceeding. |
| Proposed Solutions | Added ‘!important’ to key style properties (color, border-color, outline, box-shadow) in hover selectors in TermsPage.css to ensure the styles are applied regardless of surrounding CSS influence. |
| Follow-up Questions | 1. Does the use of ‘!important’ keyword cause conflicts or unintended overrides in other CSS files or components?<br>2. Would it be more maintainable to extract button styles into a shared CSS module or component-level style utility? |

9. Name edits in MyPage are not preserved (In Progress)

| | |
|---|---|
| Frequency | Always; whenever we try to change the name from MyPage |
| Environments | Jiwoo:<br>● MacBook Pro (Late 2023)<br>● macOS 13.5 (Ventura)<br>● Google Chrome v135.0.7049.116 (Official Build) (arm64)<br>● npm v10.8.3 |
| Severity | S2, P3 |
| Reproduction Steps | 1. Set up the development build (npm run dev)<br>2. Login, then click the “user name button” in the top right corner to go to MyPage.<br>3. Change your name from MyPage between 1 to 30 characters.<br>4. Click “Save” button. |
| Debugging Process | After changing the name, the username was not updated in the “user name button”. |
| Root Cause Analysis:<br>q. Bottlenecks<br>r. Expected impact | ● Bottleneck:<br>○ In MyPage.tsx, the name was being changed using setName() directly.<br>● Expected Impact:<br>○ If the username is not updated immediately after saving the changed name, the user could be confused if the change was successfully saved. |
| Proposed Solutions | The name was not updating due to the asynchronous nature of the setState function in React. When we call setName(name), it does not immediately update the state. The update happens after the component re-renders. So, the name was not updated immediately after calling setName(name). We are viewing the old value because the state did not update yet.<br><br>So, use useEffect function whenever the name changes. |
| Follow-up Questions | 1. Should we view the values of the set variables without re-rendering?<br>2. Are we dealing with the updated value of the set variables? |

10. DistractionsButton and FocusButton are not aligned (Resolved)

| | |
|---|---|
| Frequency | Always; whenever we come to the main page |
| Environments | Jiwoo:<br>● Macbook Pro<br>● MacOS Ventura 13.5<br>● Chrome version 135.0.7049.116 (Official Build) (arm64)<br>● Npm version: 10.8.3 |
| Severity | S3, P3 |
| Reproduction Steps | 1. Set up the development build (npm run dev)<br>2. Login, and move to mainpage |
| Debugging Process | It was clearly visible from the website. |
| Root Cause Analysis:<br>s. Bottlenecks<br>t. Expected impact | ● Bottleneck:<br>○ In MainPage.tsx, how the distraction log button is wrapped with the webcam vertically, and focus log button is wrapped in the timer, but the height is not matching could have caused the problem.<br>● Expected impact:<br>○ Since the two buttons on the bottom are not aligned, it can be viewed as not professional UX. It does not look professional to keep the buttons not aligned to each other, because it looks out of balance. |
| Proposed Solutions | We could use an inline-block display for the distractions and focus time button. And, when the size of the window reduces, we can make them in the center, putting distraction button above, and focus time button on the bottom. |
| Follow-up Questions | 1. How are we going to group the components?<br>2. Does the grouped component shown in the website looks neat and clean? |

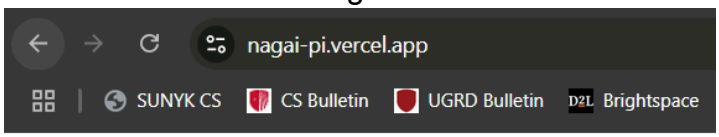
11. Logo too small, off-center & background color mismatch (Resolved)

| | |
|---|---|
| Frequency | Always; whenever we start the website in the account creation / login page |
| Environments | Jiwoo:<br>Macbook Pro<br>MacOS Ventura 13.5<br>Chrome version 135.0.7049.116 (Official Build) (arm64) |

| | |
|---|---|
| | Npm version: 10.8.3 |
| Severity | S3, P2 |
| Reproduction Steps | 1. Set up the development build (npm run dev)<br>2. Go to the account creation / login page. |
| Debugging Process | It was clearly visible from the website. |
| Root Cause Analysis:<br>u. Bottlenecks<br>v. Expected impact | ● Bottleneck:<br>○ From the AccountCreationPage.css, the accountCreation-logo image was too big, not centered, and the background color did not match.<br>● Expected impact:<br>○ Since the Account Creation Page is the first page the user would encounter our website, showing the logo image that does not give user friendly UI can make user feel like this website is not professional. |
| Proposed Solutions | ● Match the background-color<br>● Limit by max-height and max-width, with width: 230px so that the logo image would never exceed the button height and prevents it from spilling horizontally. Also, justify-content and align-items to center. |
| Follow-up Questions | 1. Whenever we put an image in our website, try to find out the optimal size of the image.<br>2. Does the official image fits well with the background color of the website? |

12. Deployed Vercel site is unresponsive & a white blank screen appears.

| | |
|---|---|
| Frequency | 2-3 times a day; usually after a prolonged period of inactivity |
| Environments | Deployment environment on Vercel & Render |
| Severity | S2, P1 |
| Reproduction Steps | 1. Navigate to the deployed website.<br>2. About approx. 6 hours after the latest backend deployment on Render, the deployed Vercel website should be unresponsive and display a blank white screen, with an API call to the 'frontend-env' endpoint |

| | |
|---|---|
| | shown as ‘Pending.’<br>  <br><br>  <br><br>3. After 1-5 minutes of constant page reloading, the site should start responding. |
| Debugging Process | I opened Developer Tools to check console and network logs. |
| Root Cause Analysis:<br>w. Bottlenecks<br>x. Expected impact | ● The ‘frontend-env’ endpoint is responsible for supplying the client with all required API keys as environment variables. Only with the secret keys loaded in the frontend environment can our app initialize; the fact that this endpoint is ‘Pending’ implies our Render backend is down.<br>○ This happens because we are currently using Render on Free Tier; there is a warning on the Render dashboard that Free Tier instances may spin down after a certain period of inactivity.<br>● With a blank, unresponsive screen displayed on the site, users may be surprised and confused. |
| Proposed Solutions | ● As we are limited to using Render on Free Tier, the best we can currently do is constantly log in to the app to prevent the backend from spinning down.<br>a. Occasionally redeploying the backend also solves the issue. |
| Follow-up Questions | 1. Will this issue be resolved if we potentially upgrade to a paid plan on Render? |