Table of Contents

Sprint 2

- Defect 1 - Chapter1 Intro Page
- Defect 2 - Chapter 1 Main Page
- Defect 3 - Chapter 1 Section 4 images
- Defect 4 - Section 4 Page (Gallery Images)
- Defect 5 - Image Gallery Component
- Defect 6 - Chapter4 Intro Page

Sprint 3

- Defect 1 - Home Page
- Defect 2 - Image Gallery Component Input validation
- Defect 3 - Image Gallery Component Keyboard Navigation
- Defect 4 - Prezi iframe Performance

Sprint 2

Defect 1 - Chapter1 Intro Page

Participants:

- **Moderator**: Man-Hua Chu (Kate)
- **Author**: Soyeon Park
- **Reader**: Soyeon Park
- **Recorder**: Sangmoon Han
- **Reviewers**: Sangmoon Han, Soyeon Park, Yu-Tse Ling (Zona), Po-Yun Hsiao (Harold)

When it Happened:

- **Date**: 27/4/2025
- **Time**: 11:05 AM

Issues Found:

1. **Artifact**: `src/frontend/src/app/chapters/chapter5/intro/page.tsx`
    - **Location**: Functionality
    - **Severity**: Medium
    - **Type**: Issue
    - **Defects Category**: Visual Representation Defects (Alternative)
    - **Description**: Consider Accessibility: If the component will include images or interactive elements in the future, consider adding accessibility features (like alt text for images).

Why This Was a Problem:

- **Missing Accessibility Fallback**: Without `alt` text, users relying on screen readers or experiencing broken image loads get no context, leading to a poor or inaccessible experience.
- **Developer Confusion**: Developers reviewing or maintaining the code can't infer the purpose of images without descriptive labels, making the interface harder to understand and debug.

Actions Taken:

- To address potential accessibility issues, Soyeon implemented alt text for all images, videos, and interactive elements. This ensures that if a media file fails to load, users will still receive contextual information about the missing content. (e.g., "Chapter 4 Introduction Image", "Quiz 2 Video", etc.), improving the usability and accessibility of the application.
- Kate confirmed that the issue was addressed, and checked the changes were properly implemented.

Why the Fix Matters:

- **Inclusive Design**: Adding `alt` text ensures all users—including those using assistive technologies—can understand what each visual element conveys.
- **Content Clarity**: It improves maintainability by giving developers clear context about each image, aiding faster debugging and future updates.

Before:

```
              {/* Teeth Image */}
              <div className='my-4 flex justify-center'>
                  <img
                      src="https://res.cloudinary.com/difs4tswt/image/upload/v1745114847/droppedImage-sı
                      className="my-4 w-1/2 mx-auto"
                  />
              </div>
```

After:

```
              {/* Teeth Image */}
              <div className='my-4 flex justify-center'>
                  <img
                      src="https://res.cloudinary.com/difs4tswt/image/upload/v1745114847/droppedImage-sı
                      alt="Teeth image"
                      className="my-4 w-1/2 mx-auto"
                  />
              </div>
```

Lessons Learned from Interactions with AI:

- **Accessibility**: From the ChatGPT review, we recognised that there are cases where users are unable to load components successfully. Alt text descriptions provide a fallback, describing what type of resource should appear at that location, enhancing accessibility.

Defect 2 - Chapter 1 Main Page

Participants:

- **Moderator**: Man-Hua Chu (Kate)
- **Author**: Soyeon Park
- **Reader**: Soyeon Park
- **Recorder**: Sangmoon Han
- **Reviewers**: Sangmoon Han, Soyeon Park, Yu-Tse Ling (Zona), Po-Yun Hsiao (Harold)

When it Happened:

- **Date**: 25/4/2025
- **Time**: 11:05 AM

Issues Found:

1. **Artifact**: `src/frontend/src/app/chapters/chapter1/chapter1Main.tsx`
    - **Location**: Variable description
    - **Severity**: Trivial
    - **Type**: Improvement
    - **Defects Category**: Visual Representation Defects (Long Line)
    - **Description**: The variable description is written in a long single line. It was suggested to break the line into multiple lines for better readability.

Why This Was a Problem:

- **Reduced Readability**: Long single-line strings are harder to scan, especially when reading or editing quickly.
- **Debugging Difficulty**: Errors or changes within a long string are more difficult to pinpoint and correct.

Actions Taken:

- Soyeon updated the code to break the long line for better readability.
- Kate confirmed that the issue was addressed, and the change was properly implemented.

Why the Fix Matters:

- **Improved Maintainability**: Splitting long lines improves readability, making the codebase easier to maintain and understand.
- **Developer Ergonomics**: Cleaner formatting supports faster debugging, collaboration, and comprehension during reviews.

Code Snippet

- Before:

```
import ChapterMain from '@/components/chapterMain';

export default function Chapter1Page() {
  const title = 'Chapter 1: An Introduction to Orthodontics';
  const description = 'Orthodontics is a highly specialized field of Dentistry, aims towards aestheti
  const sections = [
    { id: 0, title: '', href: '/chapters/chapter1/intro', icon: 'https://res.cloudinary.com/difs4tswt/
    ...
```

- After:

```
import ChapterMain from '@/components/chapterMain';

export default function Chapter1Page() {
  const title = 'Chapter 1: An Introduction to Orthodontics';
  const description =
    'Orthodontics is a highly specialized field of Dentistry, aims towards aesthetic and functional e
    'Considered first speciality of dentistry. However, rapid advancement have taken place in the las

  const sections = [
    {
      id: 0,
      title: '',
      href: '/chapters/chapter1/intro',
      icon: 'https://res.cloudinary.com/difs4tswt/image/upload/v1745402176/Intro_ChatGPT_Image_ziurdw
    }
    ...
```

Lessons Learned from Interactions with AI:

- **Readability**: Long lines, especially in variable descriptions, can make the code harder to read and maintain over time. AI helped highlight this issue early on, preventing potential issues in the future.

Defect 3 - Chapter 1 Section 4 images

Who Participated:

- **Moderator**: Yu-Tse Ling (Zona)
- **Author**: Man-Hua Chu (Kate)
- **Reader**: Man-Hua Chu (Kate)
- **Recorder**: Sangmoon Han
- **Reviewers**: Sangmoon Han, Soyeon Park, Yu-Tse Ling (Zona), Po-Yun Hsiao (Harold)

When It Happened:

- **Date**: 25/4/2025
- **Time**: 11:10 AM

Issues Found:

1. **Artifact**: `src/frontend/src/app/chapters/chapter1/section4/page.tsx`
    - **Location**: `Variable gallery1Images, gallery2Images`
    - **Severity**: Trivial
    - **Type**: Improvement

- **Defects Category**: Documentation Defects (Naming)
- **Description**: The variable names `gallery1Images` and `gallery2Images` are not descriptive enough to reflect the content of the images. It was suggested to use more descriptive names that clarify the nature or purpose of the images within the gallery.
- **Fixed by the Author?**: Yes
- **Verified by the Moderator?**: Yes

Why This Was a Problem:

- **Low Clarity**: Generic variable names like `gallery1Images` give no clue about the image content, forcing developers to dive deeper to understand their purpose.
- **Naming Collisions**: Non-descriptive, repeated patterns in naming can cause confusion and reduce the uniqueness needed for efficient search and reuse.

Actions Taken:

- Kate updated the variable names `gallery1Images` and `gallery2Images` to `extraOralIntraOralImages` and `interceptiveApplianceImages` .
- Zona confirmed that the hange was properly implemented.

Why the Fix Matters:

- **Self-Documenting Code**: Descriptive names make the code understandable at a glance, reducing cognitive load and improving onboarding for new team members.
- **Efficient Navigation**: Precise names make code easier to search and trace, especially in large files or multi-developer environments.

Lessons Learned from Their Interactions with AI:

- **Importance of Naming Conventions**: Good names serve not only the developer but also anyone who might be working with the code in the future. A name should be intuitive enough to convey the purpose of the variable or asset.

Defect 4 - Section 4 Page (Gallery Images)

Participants:

- **Moderator**: Yu-Tse Ling (Zona)
- **Author**: Man-Hua Chu (Kate)
- **Reader**: Man-Hua Chu (Kate)
- **Recorder**: Sangmoon Han
- **Reviewers**: Sangmoon Han, Soyeon Park, Yu-Tse Ling (Zona), Po-Yun Hsiao (Harold)

When it Happened:

- **Date**: 25/4/2025
- **Time**: 11:15 AM

Issues Found:

1. **Artifact**: `src/frontend/src/app/chapters/chapter1/section4/page.tsx`
    - **Location**: Variable `gallery1Images`
    - **Severity**: Trivial
    - **Type**: Improvement
    - **Defects Category**: New Functionality (Use Standard Method)
    - **Description**: The array `gallery1Images` could benefit from being dynamically generated, especially if images are stored in a more structured format or fetched from an API. The current static method for declaring the image array limits flexibility, especially when the number of images changes over time or when the data is fetched from a backend.
    - **Fixed by the Author?**: Yes
    - **Verified by the Moderator?**: Yes

Why This Was a Problem:

- **Poor Scalability**: Hardcoding the image array clutters the file and makes future updates tedious, especially as the number of images grows.
- **Lack of Modularity**: Mixing data definitions and rendering logic in the same file breaks separation of concerns, leading to less maintainable code.

Actions Taken:

- Kate updated the `gallery1Images` array to be generated dynamically. This was done by organizing the image data into a structured format and making the images easier to manage, especially as the project grows.
- The change was reviewed and verified by Zona. It was confirmed that the solution provided a more scalable approach to handling images.

Why the Fix Matters:

- **Cleaner Architecture**: Dynamically generating the array supports scalability and modularity, aligning with best practices for component design.
- **Future Flexibility**: It prepares the codebase for future updates, such as integration with backends, reducing technical debt.

Code Snippet:

- **Before**: All images are declared in `src/frontend/src/app/chapters/chapter1/section4/page.tsx`

```
export const extraOralIntraOralImages = [
  {
    src: 'https://res.cloudinary.com/difs4tswt/image/upload/v1745114790/IMG_5908-36_fbtbzy.jpg',
    caption: 'Extra Oral Frontal View Relaxed',
  },
  {
    src: 'https://res.cloudinary.com/difs4tswt/image/upload/v1745114796/IMG_5909-38_b6kls5.jpg',
    caption: 'Extra Oral Frontal View with Emotional Smile',
  },
  // more static images...
];
```

- **After**: A file called `section4Images.ts` is created to handle images.

```
export const extraOralIntraOralImages = [
    {
        src: 'https://res.cloudinary.com/difs4tswt/image/upload/v1745114790/IMG_5908-36_fbtbzy.jpg',
        caption: 'Extra Oral Frontal View Relaxed',
    },
    // more images
```

Lessons Learned from Interactions with AI:

- **Scalability and Flexibility**: This change makes it easier to modify image links and scale the content.

Defect 5 - Image Gallery Component

Participants:

- **Moderator**: Yu-Tse Ling (Zona)
- **Author**: Man-Hua Chu (Kate)
- **Reader**: Man-Hua Chu (Kate)
- **Recorder**: Sangmoon Han
- **Reviewers**: Sangmoon Han, Soyeon Park, Yu-Tse Ling (Zona), Po-Yun Hsiao (Harold)

When it Happened:

- **Date**: 25/4/2025
- **Time**: 11:30 AM

Issues Found:

1. **Artifact**: `src/frontend/src/components/ImageGallery.tsx`
    - **Location**: `images` prop validation
    - **Severity**: Medium
    - **Type**: Issue
    - **Defects Category**: Check Defects (Check User Input)
    - **Description**: There was no check to ensure that the `images` prop is a valid array and contains at least one image. It was suggested to add a check to handle cases where the `images` prop might be empty or not an array, to prevent errors during rendering.

Why This Was a Problem:

- **Crash Risk**: Without checking whether `images` is a valid array with content, the component could throw runtime errors and crash the page if the data is missing or malformed.
- **Lack of Defensive Programming**: The absence of input validation means the component assumes ideal input, which can lead to brittle behavior when integrated with real-world or evolving data sources.

Actions Taken:

- Kate added a validation check at the beginning of the component to ensure that the `images` prop is a valid array and contains at least one image. If the check fails, the component renders a message saying "No images

available".
- Zona verified that this prevents potential errors when the `images` array is empty or malformed, enhancing the component's robustness.

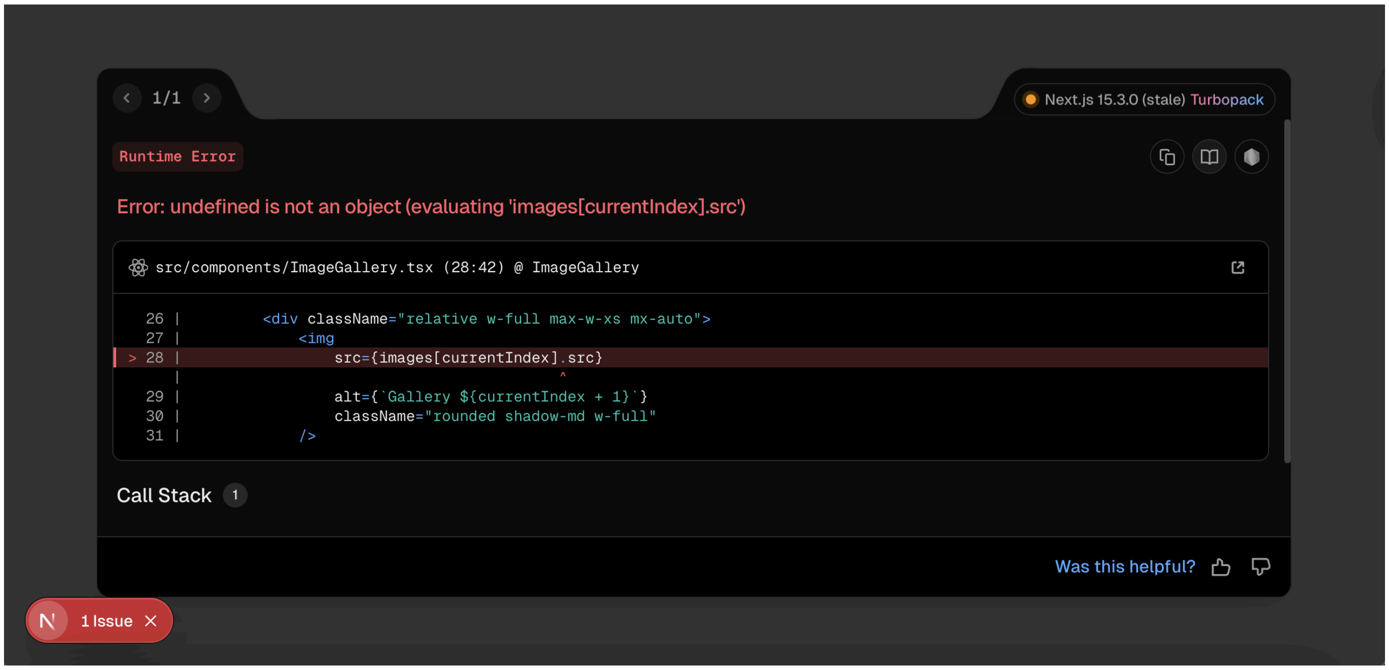
Why the Fix Matters:

- **Improved Stability**: With a validation check in place, the component can now safely handle unexpected or empty input without breaking, maintaining the user experience.
- **Better Development Practices**: Adding safeguards ensures future developers don't unknowingly introduce errors by passing incorrect props, promoting more maintainable and reliable code.

Code Snippet:

- **Before**:

```
export default function ImageGallery({ images }: Props) {
    const [currentIndex, setCurrentIndex] = useState(0);
...
}
```



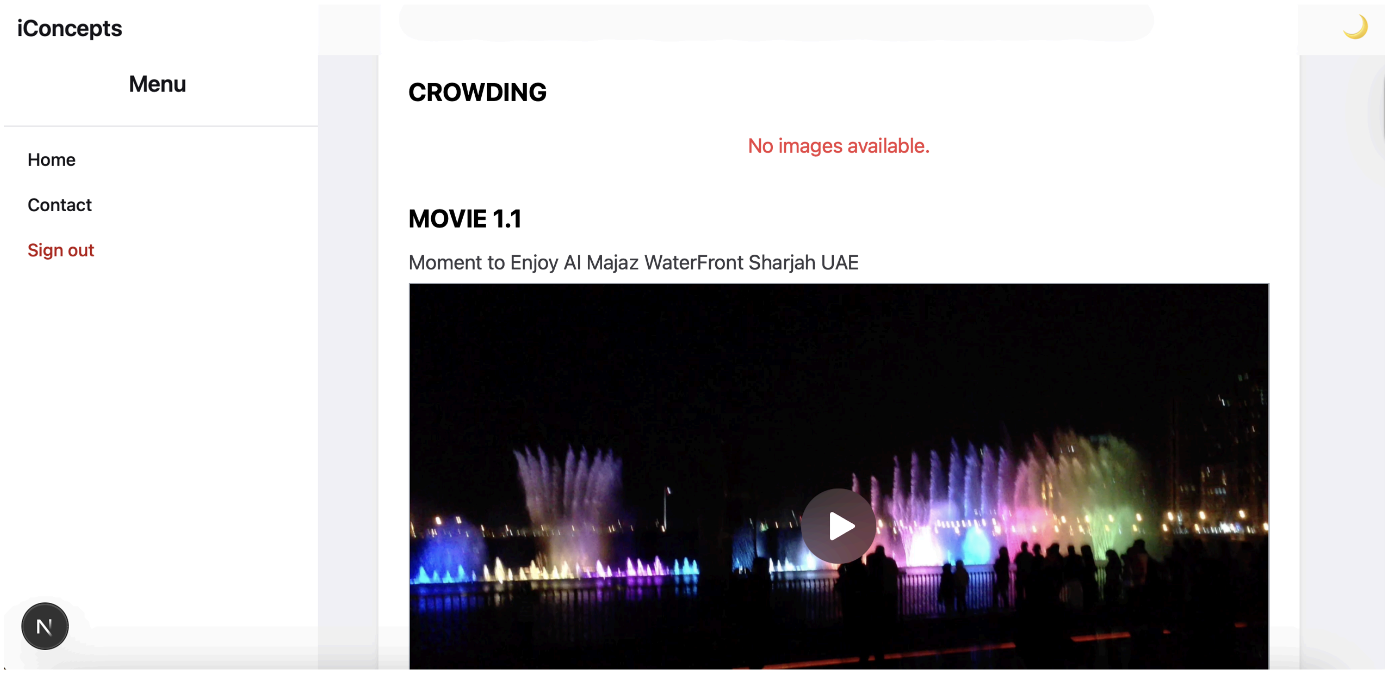
Error: undefined is not an object (evaluating 'images[currentIndex].src')

```
src/components/ImageGallery.tsx (28:42) @ ImageGallery

        <div className="relative w-full max-w-xs mx-auto">
            <img
> 28 |          src={images[currentIndex].src}
                                          ^
                alt={`Gallery ${currentIndex + 1}`}
                className="rounded shadow-md w-full"
            />
```

- **After**:

```
export default function ImageGallery({ images }: Props) {
    // Check if images is a valid array and contains at least one image
    if (!Array.isArray(images) || images.length === 0) {
        return (
            <div className="text-center text-red-500">
                No images available.
            </div>
        );
    }
    const [currentIndex, setCurrentIndex] = useState(0);
    ...
}
```


iConcepts

Menu

Home

Contact

Sign out

CROWDING

No images available.

MOVIE 1.1

Moment to Enjoy Al Majaz WaterFront Sharjah UAE

Lessons Learned from Interactions with AI:

- **Improved Code Robustness**: By implementing input validation, the component is now more robust and can handle edge cases more gracefully. It also provides better feedback to the user, making the application more reliable and user-friendly.

Defect 6 - Chapter4 Intro Page

Participants:

- **Moderator**: Man-Hua Chu (Kate)
- **Author**: Soyeon Park
- **Reader**: Soyeon Park
- **Recorder**: Sangmoon Han
- **Reviewers**: Sangmoon Han, Soyeon Park, Yu-Tse Ling (Zona), Po-Yun Hsiao (Harold)

When it Happened:

- **Date**: 27/4/2025
- **Time**: 11:15 AM

Issues Found:

1. **Artifact**: `src/frontend/src/app/chapters/chapter4/intro/page.tsx`
    - **Location**: Interface Defect
    - **Severity**: Medium
    - **Type**: Issue
    - **Defects Category**: Logic Defects (Compute)

   - **Description**: Parameter: The parameters for the `SectionPage` component are correctly passed. However, `prevHref` is set to undefined, which may not be necessary unless it is explicitly required by the `SectionPage` component.

Why This Was a Problem:

- **Broken Navigation Flow**: The absence of a valid `prevHref` made it impossible for users to navigate back from the Chapter 4 intro page to the final section of Chapter 3, creating a dead-end in the user experience.
- **Maintainability**: Leaving `prevHref` as an empty string assumes developers will know when and why it's set that way. This can lead to confusion or bugs when the project is handed over, especially if team members assume it was a mistake or placeholder.

Actions Taken:

- To address page navigation issues, Soyeon added interactive buttons for every intro pages. This ensures the usability of navigating chapters through "previous" and "next" toggle.
- Kate confirmed that the issue was addressed, and checked the changes were properly implemented.

Why the Fix Matters:

- **Improved User Experience**: The updated `prevHref` allows seamless backward navigation across chapters, ensuring users never reach a page where they cannot return to the previous one.
- **Consistency Across Pages**: This fix aligns intro pages with other sections in terms of navigation behavior, providing a uniform and intuitive experience throughout the learning material.

Before:

```
<SectionPage
    title="Chapter 4: Functional Orthopaedic Appliances"
    subtitle="Published in 2024 · Views: 19723 · Average Reading Time: 17min"
    prevHref=""
    nextHref="/chapters/chapter4/section1"
    chapterHref="/chapters/chapter1"
>
```

After:

```
<SectionPage
    title="Chapter 4: Functional Orthopaedic Appliances"
    subtitle="Published in 2024 · Views: 19723 · Average Reading Time: 17min"
    prevHref="/chapters/chapter3/section5"
    nextHref="/chapters/chapter4/section1"
    chapterHref="/chapters/chapter4"
>
```

Lessons Learned from Interactions with AI:

- **Parameter**: We organised the content by dividing it into chapters, and further splitting each chapter into multiple sections, including an introductory section for every chapter. During the initial implementation, we overlooked the navigational flow between chapters. Although the "Next" button allowed users to move forward to the introduction of the next chapter (e.g., from Chapter 3 to Chapter 4 Introduction), there was no mechanism to navigate back to the last section of the previous chapter from the introduction page.(e.g., from Chapter 4 introduction to Chapter 3)

- Through the GPT review, we identified and corrected this issue. We updated the `prevHref` parameter in the `SectionPage` component to properly link the introduction section of a chapter back to the last section of the previous chapter, ensuring seamless navigation in both directions across chapters.

Sprint 3

Defect 1 - Home Page

Participants:

- **Moderator**: Man-Hua Chu (Kate)
- **Author**: Soyeon Park
- **Reader**: Soyeon Park
- **Recorder**: Sangmoon Han
- **Reviewers**: Sangmoon Han, Soyeon Park, Yu-Tse Ling (Zona), Po-Yun Hsiao (Harold)

When it Happened:

- **Date**: 18/5/2025
- **Time**: 11:00 AM

Issues Found:

1. **Artifact**: `src/frontend/src/app/page.tsx`
    - **Location**: Whole file
    - **Severity**: Trivial
    - **Type**: Improvement
    - **Defects Category**: Documentation Defects
    - **Description**: The comments in the code are minimal. While there are some comments (e.g., // 이미지 영역 and // 탭 + 콘텐츠 영역), they are in Korean and might not be understood by all team members. It's recommended to use English comments for consistency and broader understanding.

Why This Was a Problem:

- **Understandability**: One of our key project goals is maintainability and effective collaboration. Writing comments in English—the shared working language of our team and the wider development community—ensures that code intent is clear to everyone.
- **Inclusivity**: Relying on a language only some contributors understand can lead to confusion, miscommunication, and increased onboarding time for future developers.

Actions Taken:

- Soyeon translated all Korean comments into clear English.
- She also expanded existing comments where needed to provide fuller context and better describe the purpose of each component and logic block.

Why the Fix Matters:

- As this project is intended to be handed over to another team, improving comment quality and language consistency now helps reduce friction during onboarding and ensures smoother transitions.
- Writing descriptive English comments sets a good precedent for documentation practices in future components and chapters, especially when this profect will be handover to future team(s) that may contain developers from different countries.

Before: (A snippet of the entire file)

```
{/* 탭 + 콘텐츠 영역 */}
            <div className="w-full px-4 mb-8">
                <div className="max-w-screen-xl mx-auto">
                    {/* 탭 버튼 */}
                    <div className="flex rounded-t-xl overflow-hidden">
                        {['chapters', 'videos', 'quizzes'].map((tab) => {
                            const isActive = activeTab === tab;
```

After: (A snippet of the entire file)

```
{/* Tab Navigation and Content Section */}
            <div className="w-full px-4 mb-8">
                <div className="max-w-screen-xl mx-auto">
                    {/* Tab Buttons */}
                    <div className="flex rounded-t-xl overflow-hidden">
                        {['chapters', 'videos', 'quizzes'].map((tab) => {
                            const isActive = activeTab === tab;
```

Lessons Learned from AI Interaction:

- Initially, our team overlooked the importance of writing comments for external audiences because we all understood the code and it's easier to take notes in our native languages. However, GPT highlighted the importance of considering future maintainers who may not share the same context or language background.
- This helped us recognize that documentation is not just about explaining functionality, but also about anticipating the needs of diverse collaborators—including those who will inherit our work.

Defect 2 - Image Gallery Component Input Validation

Participants:

- **Moderator**: Man-Hua Chu (Kate)
- **Author**: Man-Hua Chu (Kate)
- **Reader**: Zona Ling
- **Recorder**: Sangmoon Han
- **Reviewers**: Sangmoon Han, Soyeon Park, Yu-Tse Ling (Zona), Po-Yun Hsiao (Harold)

When it Happened:

- **Date**: 18/5/2025

- **Time**: 11:15 AM

Issues Found:

1. **Artifact**: `src/frontend/src/components/ImageGallery.tsx`
   - **Location**: ImageGallery function
   - **Severity**: Medium
   - **Type**: Improvement
   - **Defects Category**: Check Defects
   - **Description**: The `ImageGallery` component performs an initial check to ensure `images` is an array and not empty. However, it does not validate the content of each object in the array. This can lead to runtime errors if any image object is malformed or missing a valid `src` string.

Why This Was a Problem:

- **Preventing Crashes**: If `images` contains malformed objects (e.g., missing or invalid `src` values), the component can crash the entire page.
- **Robustness**: The component should be defensive and handle unexpected or faulty input gracefully, especially if data might be manipulated or modified by mistake or through external APIs.

Actions Taken:

- Kate implemented a filter to sanitize the `images` input before rendering.
- The component now checks that each item is a valid object with a `src` field that is a non-empty string. It also trims whitespace to avoid issues with improperly formatted URLs.
- Test cases were created using various forms of invalid input (e.g., empty objects, missing `src`, non-string URLs) to ensure the filtering logic works correctly.

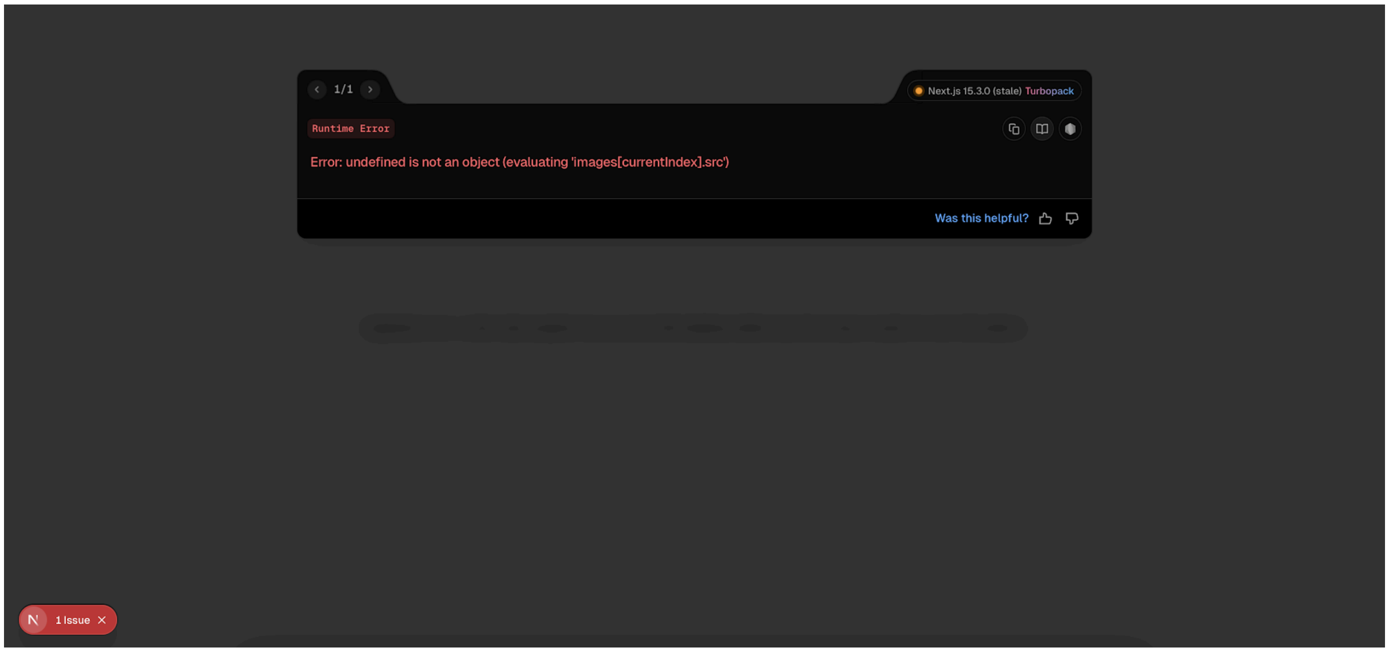
Why the Fix Matters:

- **Future-Proofing**: Although the image URLs are currently hardcoded, changes by future developers—or integration with external APIs—may introduce unexpected inputs.
- **Security & Stability**: This validation also helps mitigate any potential edge cases where data could be altered externally or maliciously, ensuring the app remains stable.

Before: (A snippet of the entire file)

```
export default function ImageGallery({ images }: Props) {
    const [currentIndex, setCurrentIndex] = useState(0);
```

❌ If the input includes invalid image objects, the app crashes:


1/1

Next.js 15.3.0 (stale) Turbopack

Runtime Error

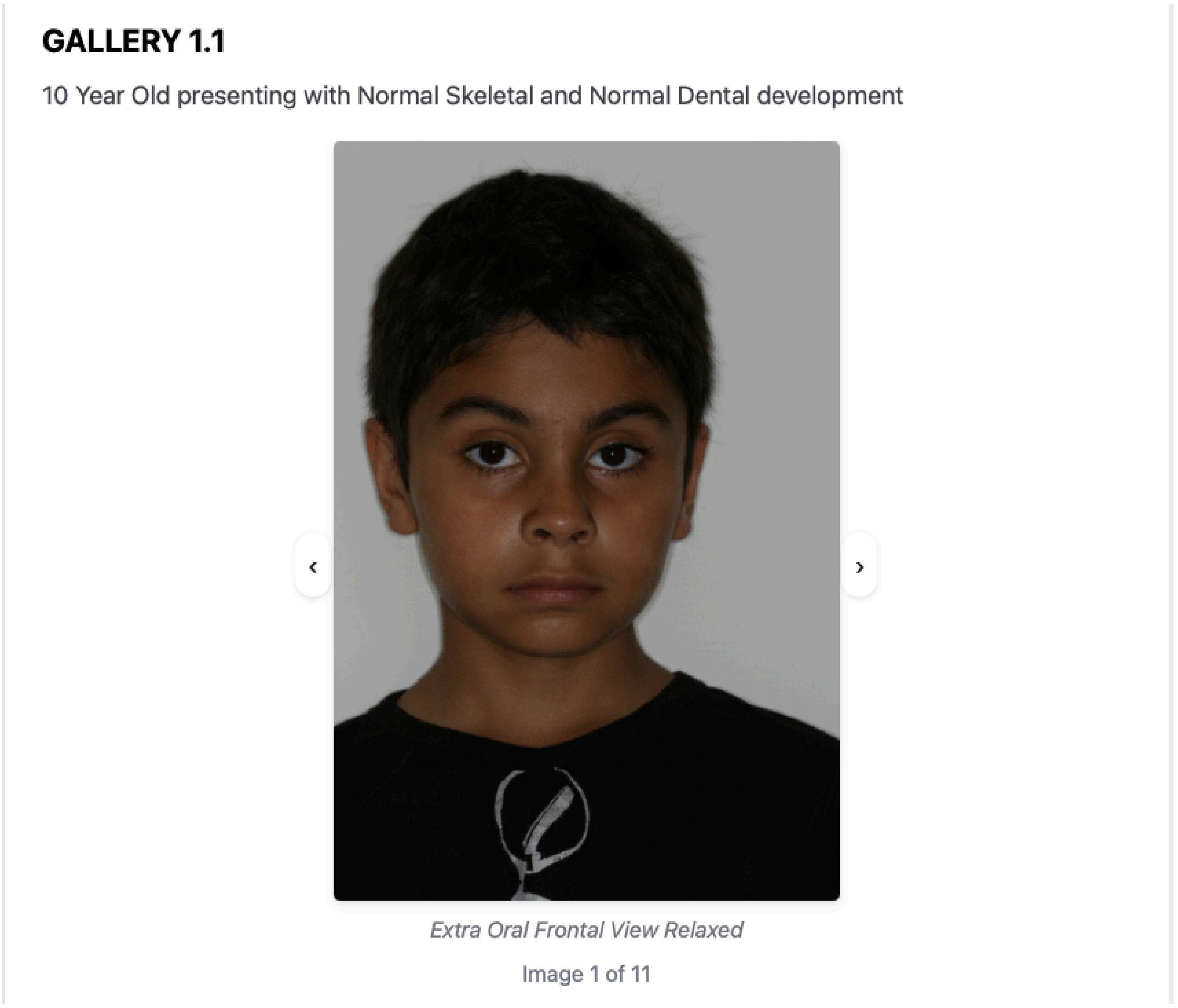
Error: undefined is not an object (evaluating 'images[currentIndex].src')

After: (A snippet of the entire file)

```
export default function ImageGallery({ images }: Props) {
    // Filter out invalid image objects
    const validImages = Array.isArray(images)
        ? images.filter(
                (img) => img && typeof img.src === 'string' && img.src.trim() !== ''
        )
        : [];

    const [currentIndex, setCurrentIndex] = useState(0);
```

✅ The component now works correctly, displaying only valid images:


GALLERY 1.1

10 Year Old presenting with Normal Skeletal and Normal Dental development

Extra Oral Frontal View Relaxed

Image 1 of 11

Lessons Learned from AI Interaction:

- We initially assumed that the image URLs were static and unchangeable, but recognized that accidents can happen—developers may modify them or data may come from dynamic sources in the future.
- This update makes the component more resilient to errors and potential malicious manipulation, ultimately improving the overall robustness of the application.

Defect 3 - Image Gallery Component Keyboard Navigation

Participants

- **Moderator**: Man-Hua Chu (Kate)
- **Author**: Man-Hua Chu (Kate)
- **Reader**: Zona Ling
- **Recorder**: Sangmoon Han
- **Reviewers**: Sangmoon Han, Soyeon Park, Yu-Tse Ling (Zona), Po-Yun Hsiao (Harold)

Date and Time

- **Date**: 18 May 2025
- **Time**: 11:30 AM

Issue Found

- **Artifact**: `src/frontend/src/components/ImageGallery.tsx`
- **Location**: `ImageGallery` function
- **Severity**: Medium
- **Type**: Improvement
- **Category**: New Functionality (Use Standard Method)
- **Description**: The image gallery originally required mouse interaction to navigate images. Keyboard navigation was not supported, limiting accessibility for keyboard users.

Why This Was a Problem:

- **Accessibility Limitation**: Users could not navigate through the gallery using a keyboard.
- **User Inconvenience**: Navigation required using a mouse or touchpad to click on buttons and scroll the page manually to bring the gallery into view.

Actions Taken

- Kate implemented keyboard navigation using the arrow keys with `useEffect`.
- Added `tabIndex={0}` to the gallery container, making it focusable.
- Styled navigation buttons to indicate focus clearly.
- Updated `alt` text for images to include captions where available.
- Tested accessibility features thoroughly using the Tab key and keyboard navigation.

Why the Fix Matters:

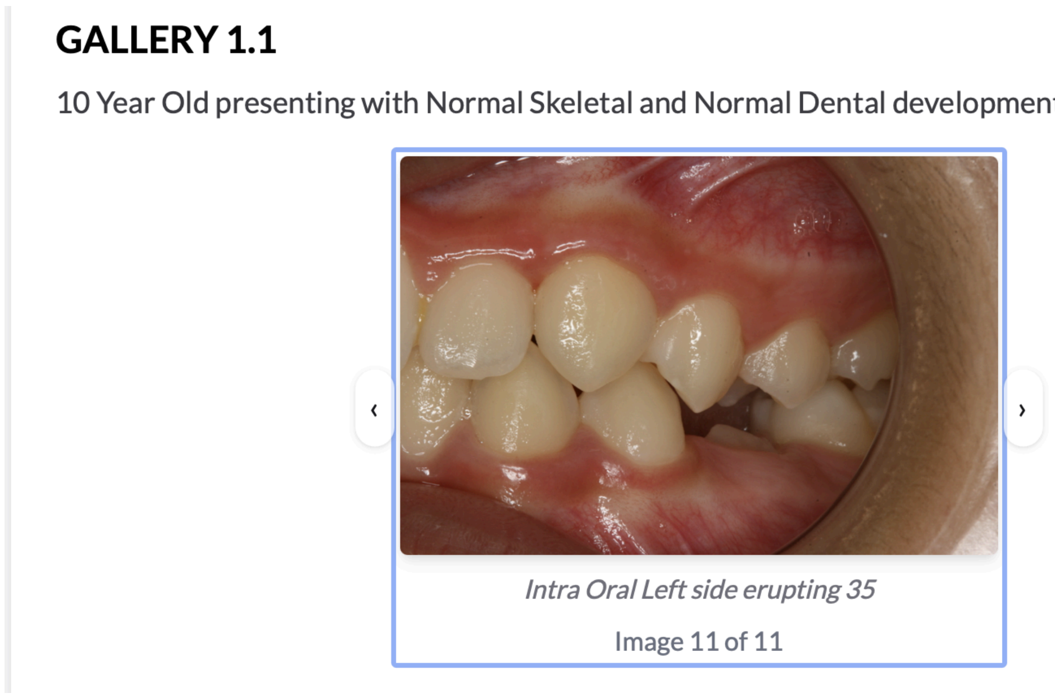
- **Improved Accessibility**: Users can now navigate the gallery using only the keyboard, making it more inclusive.
- **Enhanced Usability**: Tab key navigation brings the gallery into view quickly, improving the learning experience.
- **Robustness**: Meaningful `alt` text provides fallback information if images fail to load.

Before

- Users could only click “previous” and “next” arrows using a mouse or touchpad to switch images.

After

- Users can now:
    - Navigate images using left/right arrow keys.
    - Use Tab to focus on the gallery.
    - Access meaningful alt text for screen readers.
    - Experience better focus visibility with keyboard styling.



GALLERY 1.1

10 Year Old presenting with Normal Skeletal and Normal Dental development

Intra Oral Left side erupting 35

Image 11 of 11

Lessons Learned from AI Interaction:

- As developers, we often default to mouse-based interaction, but not all users rely on or prefer it.
- AI helped us shift perspectives and recognize the importance of keyboard accessibility.
- This led to broader thinking about inclusivity — for example, exploring multimodal support such as voice and tactile interactions to support diverse learning needs beyond visual and audio.

Defect 4 - Prezi iframe Performance

Participants

- **Moderator**: Soyeon Park
- **Author**: Soyeon Park
- **Reader**: Zona Ling
- **Recorder**: Man-Hua Chu (Kate)
- **Reviewers**: Man-Hua Chu (Kate), Sangmoon Han, Soyeon Park, Yu-Tse Ling (Zona), Po-Yun Hsiao (Harold)

Date and Time

- **Date**: 18 May 2025
- **Time**: 11:45 AM

Issue Found

- **Artifact**: `src/frontend/src/app/chapters/chapter8/intro/page.tsx`
- **Location**: Prezi iframe
- **Severity**: Trivial
- **Type**: Improvement
- **Category**: Logic Defects (Performance)

- **Description**: While performance is not currently an issue, embedding external Prezi content via iframe may impact page load times. The team considered optimizing it using lazy loading to improve performance.

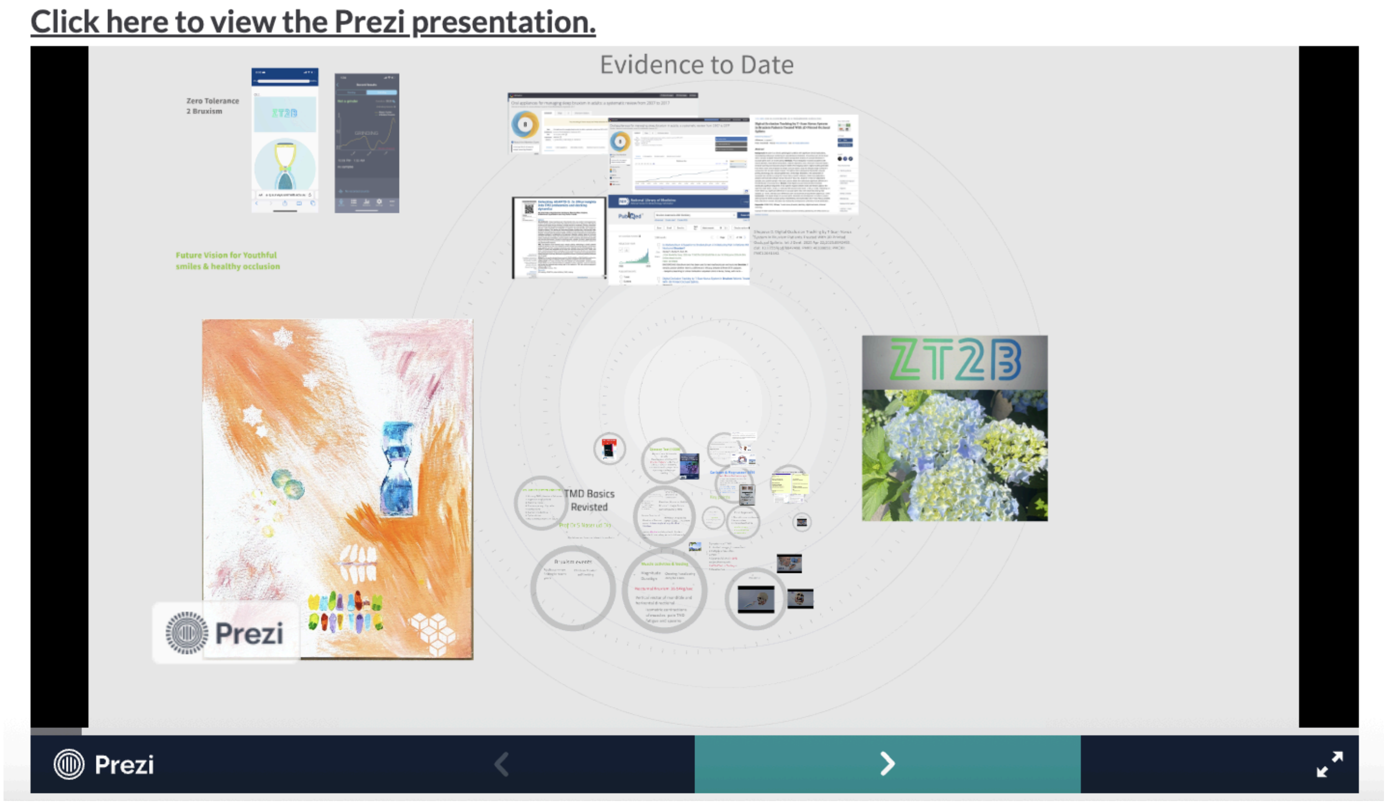
Why This Was a Problem:

- **Performance Concern**: Prezi is an interactive external content platform and hosted outside our main content delivery system (Cloudinary). Its iframe might delay initial rendering or loading metrics.
- **Technical Constraint**: Migration to Cloudinary or internal hosting isn't feasible due to client-side dependencies and ownership of the Prezi content.

Actions Taken

- Soyeon followed the suggested Solution, enabling `loading="lazy"` on the iframe to defer loading until it enters the viewport.
- Kate Conducted performance tests before and after using Chrome DevTools' Performance tab.
- Metrics Analyzed:
  - LCP (Largest Contentful Paint): Measures the render time of the largest visible content block.
  - CLS (Cumulative Layout Shift): Measures visual stability (how much layout shifts unexpectedly).
  - INP (Interaction to Next Paint): Measures how quickly the page responds to user input.

Prezi component


Click here to view the Prezi presentation.
Evidence to Date
Zero Tolerance 2 Bruxism
Future Vision for Youthful smiles & healthy occlusion
TMD Basics Revisited
ZT2B
Prezi
Prezi

Performance Results

Metric	Before	After	Target	Remarks
LCP	2.64 s	3.36 s	≤ 2.5 s (Good), ≤ 4 s (Needs Improvement)	Lazy loading increased LCP unexpectedly

Metric	Before	After	Target	Remarks
CLS	0	0	≤ 0.1	No layout shift — result is good
INP	20 ms	32 ms	≤ 200 ms	All ≤ 200 ms, which is good.

Decision and Rationale

Since the LCP worsened and CLS remained unaffected, the team decided to **revert to the original non-lazy-loading implementation**. The lazy-loading optimization, in this context, did not produce the expected performance gain and even introduced delays.

Lessons Learned from AI Interaction

- AI can highlight potential issues and suggest common optimizations.
- However, not all suggestions are context-appropriate. Testing and validation are essential before adopting any fix.
- Critical evaluation of AI feedback ensures development decisions are data-driven and not based solely on generic best practices.