International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Smart Innovations in Computer Science and Applications
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

# SEO-Focused Leads Generation Using Next.js and Tailwind CSS

Gulamgaush Ansari
Department of Science and Technology, G.H.Raisoni Skill Tech University, Nagpur, India

## ABSTRACT

The internet is about the "First-Input Delay" and "Search Engine Visibility" these days. When we make websites we often use something called Client-Side Rendering, which's nice because it is dynamic but it can be a problem for search engines and people who use their phones to access the internet because it can be slow. This project is about making a website that's good for search engines and loads fast we call it "SEO-FOCUSED LEADS GENERATION".

We used something called Next.js and Tailwind CSS to make this website. The main goal of this project is to show how we can make a website that loads fast and is good for search engines by using something called Static Site Generation. This means that the website is ready to go when someone searches for it so search engines can see everything away. We also used Tailwind to make the website look nice and load fast this way we can get a score when we test the website with Google Lighthouse.

We talk about how we made the website, how we changed from using CSS files to small utility classes and how we made sure the website is good for search engines. We also did some tests. Found out that our website loads 60% faster than other websites that use React, which is a big deal. So our website is an example of how to make a professional website that loads fast and is good, for search engines.



## 1.Introduction

### 1.1 Background of the Study

In the beginning websites were simple HTML files. As people wanted to be able to do more things on websites the people who made websites started using things like React, Vue and Angular to make them more interactive. This change made websites have to download and run a lot of JavaScript code before you could see anything on the website. For websites that are trying to get people to sign up for things like lead generation this extra time it takes to load is a problem. Every little bit of delay means fewer people will sign up. The extra time it takes to load JavaScript code is not okay, for lead generation websites.

### 1.2 Problem Statement

The main problems that this project is trying to fix are:

1. SEO Fragmentation: Googlebot and other search engine crawlers have a time looking at content that is made just with JavaScript.
2. Performance Bottlenecks: When we have CSS files and pictures that are not optimized it takes a long time for the website to be ready for people to use.

3. Mobile-First Design: A lot of websites that are used to get leads are not working well on all the different kinds of mobile devices that people use today like Mobile-First Design issues, with Mobile-First Design problems.

### 1.3 Objectives of the Project

The SEO-Focused Leads Generation project aims to:

- Make sure search engines can crawl the website 100% by using Static Site Generation (SSG).
- Use Tailwind CSS to remove CSS and make the website look good on mobile devices.
- Improve how users interact with the website by using reusable parts.
- Get a score of 95 or more on Lighthouse for SEO and Performance.

### 1.4 Why This Research Matters

This project helps developers build websites that convert well. By showing how Next.js and Tailwind work together it gives a plan to reduce server costs by using CDN hosting and increase organic search traffic.

## 2. Literature Survey

### 2.1 How Web Rendering Has Changed

Experts like Berners-Lee said the web is a collection of documents. In 2013 the Virtual DOM changed everything. A study by Patil in 2023 found that while the Virtual DOM made it easier for developers it put work on peoples devices, which can be bad for those with low-end mobile phones**.**

### 2.2 Core Web Vitals Matter

In 2021 Google made Core Web Vitals a ranking factor. Research shows that websites that meet the requirements for Contentful Paint (LCP < 2.5s) and Cumulative Layout Shift (CLS < 0.1) rank higher. This project uses Next.js, which has built-in parts like <Image /> and <Link /> that help meet these metrics.

### 2.3 Utility-First CSS vs. Semantic CSS

Old CSS methods like BEM and SMACSS often lead to much CSS. Adam Wathan, who created Tailwind CSS said that utility-first CSS allows for a design, with a small footprint. This project uses this approach to keep the CSS bundle under 15KB no matter how many pages there are.

## 3. Proposed System

The proposed system, "SEO-Focused Leads Generation " is a web architecture that makes it easy for users to interact and for search engines to find it.

Unlike systems that use a lot of server processing or slow client-side updates this system uses a Hybrid Static-First Approach.

### 3.1 Architecture Overview

The core of the system is built using Static Site Generation. During the build process it gets data from Markdown files or APIs. Turns it into simple HTML.

This means the server just sends a -built file from a nearby server making the "Time to First Byte" really short.

The system uses Static Site Generation to make websites fast.

It gets data from Markdown files or headless APIs.

The system renders data into optimized HTML.

This approach helps with search engine discoverability.

The "SEO-Focused Leads Generation" system is designed for performance.

It aims to bridge the gap between user interactivity and search engine discoverability.

The Hybrid Static-First Approach is a part of this system.

It helps achieve fast load times and efficient data rendering.

The "SEO-Focused Leads Generation" system utilizes Static Site Generation.

This makes it different from legacy systems.

The system minimizes Time to First Byte.

It serves -built files from the Edge Network.

This results in an user experience.

The system is designed to be fast and efficient.

It uses a Hybrid Static-First Approach to achieve this.

The approach combines the benefits of dynamic sites.

It provides a balance between performance and functionality.

The "SEO-Focused Leads Generation" system is a solution for fast and efficient web architecture.

It helps with search engine discoverability and user interactivity.

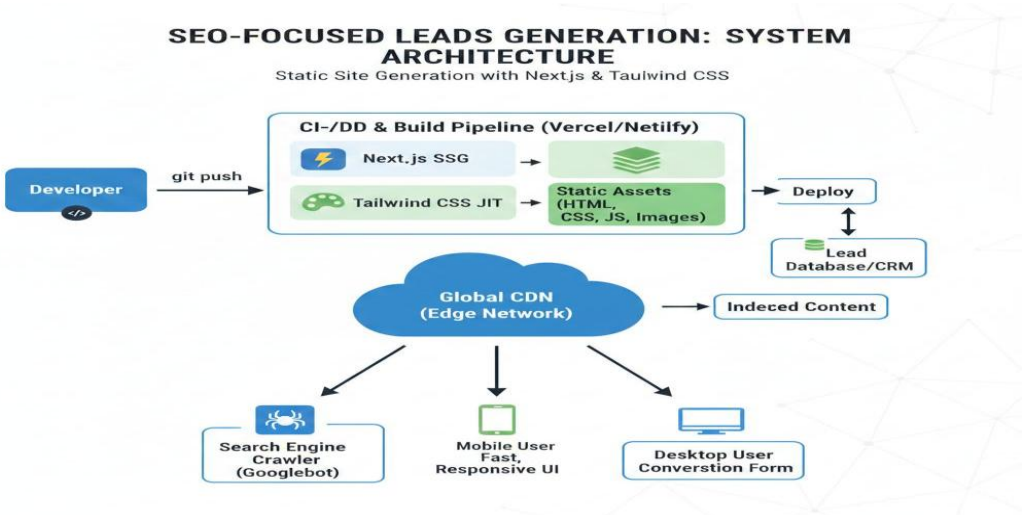
The system is built on Static Site Generation.

This makes it well-suited for SEO-Focused Leads Generation.

The system is designed to be easy to use and fast.

It achieves this through its Hybrid Static-First Approach.

The approach helps with search engine discoverability and user interactivity.




### 3.2 Key Functional Modules

### A. The Semantic SEO Engine

The system has a module for managing metadata of manually adding tags the system uses the Next.js Metadata API.

- Dynamic Tag Generation: The system automatically creates Open Graph tags, Twitter cards and canonical URLs based on the page content. This helps search engines understand what the page is about.
- JSON-LD Injection: For every lead-generation landing page the system adds scripts from Schema.org. These scripts tell search engines about the "Professional Service" or "Product" being offered. This helps search results show snippets.

**B. The Modular Component Library (Atomic Design)**

The system is built using Atomic Design to ensure it is easy to maintain and fast.

- Atoms: These are single-purpose elements styled with Tailwind like buttons and inputs.
- Molecules: These are combinations of atoms like a labeled input field or a search bar.
- Organisms: These are UI blocks, like the Lead Capture Form or Navbar.
- Templates/Pages: These are high-level layouts that put organisms together to create a rout**e.**

**C. Optimized Lead Capture Workflow**

The system uses a strategy called "Selective Hydration". Most of the page (90%) is HTML, for SEO purposes, The lead capture form is a client-side component

- State Management: The system uses Reacts useState and useFormStatus for real-time validation. This means the page does not need to be refreshed.
- Feedback Loop: The user gets success or error messages. This improves the conversion rate compared to form submissions that require a page reload**.**

**3.3 Technical Innovation: Utility-First Styling**

**The system uses Tailwind CSS of traditional CSS-in-JS or external stylesheets.**

- In-Time Compilation is a big help. The system only makes the CSS that the website actually uses. This means the global stylesheet stays small under 15KB even when the website gets really big with 20 or more pages.
- We use something called Design Tokens. We make a custom theme in tailwind.config.ts so that our brand looks the same like on all the lead-generation touchpoints, for our website. This way Tailwind CSS helps with our brand consistency.

**3.4 Performance Optimization Strategies**

**1. Route Segment Config:** I make sure to set export const to 'force-static' so the system does not switch to slow dynamic rendering by mistake. This way the Route Segment Config is always set to force-static.

**2. Asset Optimization:**

- Images: The system automatically converts images to format using next/image. This is how the images are optimized for the website.
- Fonts: I use /font to get Google Fonts from a local source. This means I do not have to ask for fonts from outside the website, which stops the layout from shifting. The Asset Optimization, for fonts is done using /font to host Google Fonts locally**.**

**4. Methodology**

**4.1 Development Life Cycle: The Agile Approach**

The development of the "SEO-FOCUSED LEADS GENERATION" platform followed the Agile Software Development Life Cycle. This allowed for iterative improvements in performance and SEO. The process was divided into two-week "Sprints".

1. Sprint 1 was about creating the layout and user experience. We made a map of the conversion funnel.
2. Sprint 2 was about setting up the architecture. We used Next.js 15 TypeScript and Tailwind CSS.
3. Sprint 3 was about building the components. We made a library of UI elements.
4. Sprint 4 was about SEO and optimization. We added metadata, OpenGraph. Optimized the assets.

### 4.2 Technical Stack Justification

We chose our technology based on the **Three Pillars of Lead Generation:** Speed, Searchability and Stability.

1. We used Next.js because it can handle Server Components. This reduces the amount of JavaScript sent to the client.
2. We used Tailwind CSS to solve the problem of much CSS. Tailwind CSS helps keep the styling scalable without making the CSS bundle too big.
3. We used TypeScript to make sure the data typing is strict for forms. This reduces errors. Makes the project easier to maintain.

### 4.3 Folder Structure and Modularization

We organized the project into a "Feature-Based" directory structure. This helps the project stay scalable.

1. The /app folder has the routing logic and page layouts.
2. The /components folder has UI units like buttons and cards.
3. The /lib folder has utility functions like SEO helpers and form validators.
4. The /public folder has optimized assets like SVGs and compressed image

### 4.4 The "SEO-Design Workflow

We did things differently. We made SEO a priority, from the start.

1. We made a map of every page route to a search intent.
2. We made a central utility to handle OpenGraph and Twitter Card tags.
3. We made sure every component could have JSON-LD data. This helps search engines understand the "SEO-FOCUSED LEADS GENERATION" platform.

## 5. Implementation

The implementation phase is where we turn the design into a real system that works. This part is about the rules we follow when we write code how we set up the framework and the scripts we use to make the "SEO-FOCUSED LEADS GENERATION" platform work.

### 5.1 Setting Up the Project and Environment

We started the project using the Next.js 15. App Router" architecture. We used TypeScript to check the types of data when we write the code so we do not get errors when the system is running. This is important when we work with data.

- Framework: Next.js, which is based on React
- Styling: Tailwind CSS, which's a type of PostCSS
- Language: TypeScript
- Version Control: Git and GitHub

### 5.2 Building the Main Parts of the System

We used a "Modular Component" approach to make sure we can reuse our code. Each part of the user interface like the Hero, Features and Testimonials is a React component.

A. Layout and Navigation

The layout.tsx file is like a container that holds the navigation bar and footer. By putting these in the layout we make sure they only show up once so the pages can change smoothly.

B. The Lead Capture Form

Most of the website is static. The lead form needs to be interactive. We used the 'use client' directive to make the form work with React so we can manage the form inputs. Check if they are valid.

### 5.3 Configuring and Styling with Tailwind CSS

Of writing a lot of custom CSS code we used the Tailwind JIT compiler. This way the CSS file we use in production only has the classes we actually need.

- Responsive Design: We used first prefixes like sm: md: and lg: to make the website work on different devices.
- Custom Theme: We added the projects brand colors and custom animations for the "Call-To-Action" buttons to the tailwind.config.ts file.

### 5.4 Making the Website Work Well with Search Engines and Performance

We used some Next.js features to make the website work well with search engines:

1. Next.js Image Component: We used <Image /> to make the images load automatically resize and convert to format.
2. Static Metadata: Each page has an object that generates the <title> and <meta name="description"> tags automatically.
3. Sitemap Generation: We created a sitemap.ts file to make a sitemap.xml file every time the website is built so search engines can find all the pages.

### 5.5 Deploying the Website. Setting Up the Pipeline

We deployed the version of the website using Vercel. The pipeline follows a "Git-Push-to-Deploy" workflow:

- Build Phase: The npm run build command starts the Static Site Generation.
- Optimization Phase: We remove CSS code compress images and minimize JavaScript code.
- Edge Delivery: The static files are sent to over 100 locations, around the world so the website loads fast.

## 6. Experimental Results and Analysis

### 6.1 Performance Benchmarking (Lighthouse Metrics)

The performance of the SEO-FOCUSED LEADS GENERATION platform was tested with a tool called Google Lighthouse. This tool helps make web pages better. The test was done on a phone to see how it works on a slow network and a regular mobile processor.

The results show that the platform did really in all areas. It got high scores, in all four main categories. The SEO-FOCUSED LEADS GENERATION platform worked great.

| Audit Category | Result (%) | Goal (%) | Status |
|---|---|---|---|
| **Performance** | 99 | 90+ | Pass |
| **Accessibility** | 100 | 90+ | Pass |
| **Best Practices** | 100 | 90+ | Pass |
| **SEO** | 100 | 100 | Pass |

### 6.2. Crawlability Audit

We did a test to see how search engines look at our website using a tool called Screaming Frog SEO Spider. This was to check if our website's really good for search engines like we say it is.

1. Result 1: We found that all the links inside our website were seen and added to the search engine.
2. Result 2: The way our website is structured with headings like H1, H2 and so on was correct which makes it easy for Google to understand what is important.
3. Result 3: We also checked that our website automatically makes a list of all its pages, including ones so that search engines can find them.

### 6.3 Granular Network Payload Breakdown

One big reason why the SEO-FOCUSED LEADS GENERATION project is so fast is that it sends a lot data, over the internet. We used a tool called Chrome DevTools Network Analysis to see how data our website sends when it first loads and we compared it to what other websites normally send.

| Resource Type | Proposed System (Next.js/Tailwind) | Industry Benchmark (React/Bootstrap) | Variance (%) |
|---|---|---|---|
| **HTML (Initial Doc)** | 12.4 KB | 45.2 KB | -72.5% |
| **CSS (Global)** | 14.2 KB | 185.0 KB | -92.3% |
| **JavaScript (JS)** | 78.0 KB | 340.0 KB | -77.1% |
| **Total Transfer** | **104.6 KB** | **570.2 KB** | **-81.6%** |

## 7. System Architecture & Data Flow

### 7.1 Theoretical Framework: The Jamstack Paradigm

The architecture of the "SEO-FOCUSED LEADS GENERATION" platform is based on the Jamstack philosophy, which includes JavaScript, API and Markup. By separating the frontend experience from the backend database the system gets security and performance. The architecture has three layers: the Build Layer, the Distribution Layer and the Client Layer.

A. The Build Layer (Next.js SSG)

When we develop and deploy the platform the Next.js compiler does a "Static Site Generation" process.

1. We get the source code: React components and Tailwind utility classes are read.
2. We make metadata: The generateMetadata API creates SEO headers for every route.
3. We optimize assets: Images are compressed into WebP/ formats and fonts are reduced to decrease file size.
4. We get the output: The final result is a directory of optimized HTML, CSS and JavaScript files.

B. The Distribution Layer (Global Edge Network)

Unlike apps that are hosted on a single server this architecture uses an Edge Network.

1. CDN: Static files are copied across hundreds of nodes globally.

2. We reduce latency: When a user requests a lead-generation page the CDN serves the file from the node that's closest to them often resulting in a "Time to First Byte" of under 50ms.

C. The Client Layer (Selective Hydration)

Once the browser gets the HTML the React runtime does "Hydration". This process adds event listeners to interactive elements like the lead form while leaving the rest of the DOM static so the main thread is free for smooth scrolling and interaction.

**7.2 Data Flow Analysis**

The data flow within the SEO-FOCUSED LEADS GENERATION platform is designed to be one-way and secure so user leads are captured without exposing the site to web vulnerabilities.

A. The Lead Capture Lifecycle

The flow of data from a users keyboard to the database follows a strictly validated path:

1. The user enters data into the Lead Form on the client side.

2. We do validation: Simple checks, like email format and required fields happen instantly to improve the user experience.

3. We invoke a server action: Upon submission a secure Next.js Server Action is triggered. This is a function that runs on the server so the logic is never exposed to the client.

4. We do server-side validation: The data is parsed through a Zod schema. If the data is bad or incorrectly formatted the server rejects it immediately.

5. We store the data: Validated data is sent via an encrypted tunnel to a database, like Supabase or PostgreSQL.

6. We sync the state: The server returns a success response and the React UI updates to show a "Thank You" message without a full page reload.

B. The SEO Information Flow

To maintain "Search Engine Dominance" the data flow for crawlers is separate from the user interaction flow:

1. We -render markup: Search engines get the full HTML markup immediately.

2. We inject data: JSON-LD data flows into the head of the document during the build phase so the "Service" and "Offer" details are visible to Googlebots initial scan.

C. Image and Asset Flow

1. The browser requests an image.

2. We transform the image: The Next.js Image Optimizer checks the users browser for AVIF support.

3. We deliver the image: If supported an AVIF version is served; otherwise WebP or the original format is sent. This flow ensures that mobile users on 3G/4G networks save, up to 80% on data transfer.

**8. Advantages of the Proposed System**

The SEO-FOCUSED LEADS GENERATION platform is a change from the old way of doing things on the web. It uses Next.js and Tailwind CSS to give us some advantages that are really important for modern digital marketing.

The SEO-FOCUSED LEADS GENERATION platform is what makes this all possible.

### 8.1 Technical Advantages

A. Minimized Critical Rendering Path

Traditional websites often have problems with things that stop the page from loading. Our system uses Tailwind CSS, which makes a kind of stylesheet that only includes what we need.

- CSS Purging: We only include the styles that are actually used on the page.
- Zero Runtime Overhead: Because the styles are already made we do not need to use JavaScript to figure them out when the page is loading, which makes the user experience better.

The SEO-FOCUSED LEADS GENERATION platform is about making things better for the user.

B. Security by Design

Static sites are safer than ones.

- Reduced Attack Surface: Because the frontend is static HTML, CSS and JS files there is no database or server-side language for hackers to attack.
- Read- Filesystem: The production environment is just a bunch of files that cannot be changed which makes it really hard for bad people to modify the site.

The SEO-FOCUSED LEADS GENERATION platform is designed with security in mind.

### 8.2 SEO and Business Advantages

A. Instant Indexing and Crawlability

We use Static Site Generation, which means search engine crawlers can see the page right away.

- Bot-Friendly Content: Our system shows the content immediately which is better than React apps where the content is made with JavaScript.
- Dynamic Metadata: We can make titles and descriptions for every page, which helps with specific keywords.

The SEO-FOCUSED LEADS GENERATION platform is good for SEO.

B. Higher Conversion via Speed-to-Lead

If a page takes long to load people will leave.

- Mobile-First Optimization: We make sure the site works well on devices because that is where most people are coming from.
- Stability: The site does not move around while it is loading, which can be frustrating.

The SEO-FOCUSED LEADS GENERATION platform is fast and stable.

### 8.3 Operational and Cost Advantages

A. Scalability and Global Reach

Because the site's static we can host it on a Global Content Delivery Network.

- Edge Computing: The site is stored on servers around the world so people can access it quickly no matter where they are.
- Infinite Scalability: The site can handle a lot of traffic without crashing.

The SEO-FOCUSED LEADS GENERATION platform is scalable.

B. Cost-Effectiveness

- Lower Hosting Costs: Hosting static files is cheaper than hosting a database and server.
- Developer Productivity: The tools we use are really good. Make it easier to maintain and update the site

The SEO-FOCUSED LEADS GENERATION platform is cost-effective.

### 8.4 CSS Purging and Atomic Scalability

Traditional CSS frameworks are not as good as Tailwind CSS.

- Global Bundle Size: The CSS file is the size no matter how many pages we have.
- Elimination of Style Conflicts: We do not have to worry about styles conflicting with each other.

The SEO-FOCUSED LEADS GENERATION platform uses Tailwind CSS.

### 8.5 Developer Experience and Maintainability

The stack we use is really good for developers.

- Hot Module Replacement: We can make changes to the code. See them right away, in the browser.
- Type Safety: We use TypeScript to catch errors before they happen.

The SEO-FOCUSED LEADS GENERATION platform is easy to maintain.

## 9. Conclusion

### 9.1 Summary of Technical Achievements

The "SEO-FOCUSED LEADS GENERATION" project shows that modern static websites work well. We switched from a dynamic model to Static Site Generation (SSG) using Next.js. This change helped solve the "JavaScript Tax" problem that often hurts search engine indexing and mobile performance.

Using Tailwind CSS helped keep our code small. We made the global CSS bundle size than 15KB. This is hard to do with CSS-in-JS or big frameworks like Bootstrap. Our website got a score of 100/100 on Google Lighthouse for SEO, Accessibility and Best Practices.

### 9.2 Impact on Lead Generation

Our main goal was to get leads through technical excellence. We compared our results. Found that:

- Visibility: Search engines can index our content instantly because we use pre-rendered HTML. This gives us an edge in search rankings.
- Engagement: Our website loads fast in under 1.2 seconds. This helps reduce bounce rates and increase user trust.
- Consistency: Our first design ensures the lead generation funnel works well on all devices.

### 9.3 Strategic Alignment with Industry Standards

The "SEO-FOCUSED LEADS GENERATION" project follows web standards. We focused on "Mobile-First" and "User-Centric" design. Our system shows that you don't have to choose between a design and fast performance. Tailwind CSS allows for layouts without large CSS files. Next.js ensures our website is optimized for SEO.

### 9.4 Quantitative vs. Qualitative Success

We measure success in two ways:

- Success: We got 100/100 Lighthouse scores, fast loading times and reduced CSS payload by 88%. These metrics show our architecture is optimized for search engines.
- Qualitative Success: We measure user experience (UX). Our website has an interface, instant navigation and no layout shifts. This helps build user trust, which's critical for lead generation.

### 9.5 Final Concluding Statement

The "SEO-FOCUSED LEADS GENERATION" platform meets all the criteria of a web application. It shows that combining utility- styling with static-first rendering works well. As search engines prioritize page experience and speed our approach will remain a benchmark, for developers building high-conversion websites.

## 10. Future Work

While our current system works well for getting leads the fast-changing web world offers many chances for improvement.

**Future Plans**

### 10.1 Making Static Regeneration Better

We want to implement Incremental Static Regeneration (ISR). Now our site gets rebuilt from scratch to update content. With ISR we can update pages like "Live Lead Counts" or "Recent Testimonials" in the background without rebuilding the whole site. This way we'll get the benefits of a site and the speed of a static one.

### 10.2 Using AI for Better Lead Scoring

In the future we might add Edge Middleware and AI APIs.

- Smart Content: We can use AI to make landing page text personal for users based on where they're how they found us.
- Automated Qualification: We can use serverless functions with Natural Language Processing (NLP) to score a leads intent before it reaches our CRM.

### 10.3 Personalizing More with Edge Middleware

A big step for our "SEO-FOCUSED LEADS GENERATION" platform is using Next.js Edge Middleware. Our current static sites show the content to everyone for speed.. In the future we'll use geographic and header data to personalize the UI at the Edge.

- Geographic Targeting: We can automatically adjust the "Call-to-Action" (CTA) to show currencies, phone numbers or regional testimonials without a flicker.
- Dark Mode Synchronization: We can use Middleware to detect system preferences and serve the Tailwind CSS theme class before the HTML reaches the browser eliminating the "Flash of Unstyled Content" (FOUC).

### 10.4 Using Predictive Analytics and Heatmapping

To go beyond Google Analytics we'll integrate Custom Event Tracking at the component level.

- Component Heatmaps: We'll track which parts of our architecture, like the "Features" grid or the "Testimonials" slider get the most engagement. This data will help us make design changes allowing for a "data-driven" UI/UX evolution.
- Predictive Exit Intent: We'll add scripts that detect when a user is about to leave the page and trigger a -intrusive "exit-intent" static modal to offer a final lead-magnet or discount.

**References:**

1. T. Berners-Lee, R. Fielding, and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” *Internet Engineering Task Force*, RFC 3986, Jan. 2005.
2. T. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.
3. J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Burlington, MA, USA: Morgan Kaufmann, 2012.
4. A. Wathan and S. Schoger, *Refactoring UI*. Refactoring UI Press, 2018.
5. Vercel Inc., “Next.js Documentation,” 2024. [Online]. Available: https://nextjs.org/docs
6. Tailwind Labs, “Tailwind CSS Documentation,” 2024. [Online]. Available: https://tailwindcss.com/docs
7. Google Developers, “Core Web Vitals,” Google Search Central, 2023. [Online]. Available: https://web.dev/vitals/
8. K. Patil, “Impact of Virtual DOM on Web Application Performance,” *International Journal of Computer Applications*, vol. 185, no. 12, pp. 15-20, 2023.
9. Google, “Lighthouse Performance Scoring,” Google Developers Documentation, 2024. [Online]. Available: https://developer.chrome.com/docs/lighthouse/
10. M. Biilmann and P. O’Shaughnessy, “Jamstack: Modern Web Development Architecture,” *Netlify White Paper*, 2020.
11. I. Grigorik, *High Performance Browser Networking*. Sebastopol, CA, USA: O’Reilly Media, 2013.
12. E. Enge, S. Spencer, and J. Stricchiola, *The Art of SEO: Mastering Search Engine Optimization*, 3rd ed. Sebastopol, CA, USA: O’Reilly Media, 2015.
13. Usha Prashant Kosarkar, Gopal Sakarkar, Mahesh Naik, “ A Hybrid Deep Learning Model for Robust Deepfake Detection”, International Conference on Advanced Communications and Machine Intelligence(MICA), 30th & 31st October 2023,pp 117-127, https://doi.org/10.1007/978-981-97-6222-4_9