# What are the best practises in eCommerce?

## Contents

# Preface

For my individual project I have lots of freedom in the technical choices. However, every technical choice that I make must be validated with sources to prove it is indeed a good decision. To research what choices I should make, I looked for the most important best practises in the context of eCommerce. Here are the most important non-functional requirements according to (Technologies, 2021): **Performance, Scalability**, **Security, Maintainability**, **Usability.** These non-functional requirements also allow me to touch all the learning outcomes.

# How can I optimize page loading speed?

In today's competitive e-commerce landscape, page load speed is a critical factor in user satisfaction and conversion rates. According to research about page loading speeds (Wiegand, 2022) an e-commerce website must load within 1 second or less to retain potential customers effectively. Websites that take longer than this threshold risk losing a significant portion of their visitors, as users tend to abandon pages that are slow to load.

Given the importance of swift page loading, this research aims to explore and identify key strategies for optimizing response times in e-commerce applications. By focusing on techniques and best practices that address common performance bottlenecks, this study seeks to provide actionable insights that can help me achieve faster loading speeds and improve overall user experience. This research will cover critical optimization strategies, including image and media management, caching techniques, hosting performance, and the reduction of render-blocking resources.

## Best practices

According to research on page speed optimization (Schaff, 2019) and (MOZ, 2021), several key strategies can significantly enhance your website's performance. The following table outlines these essential optimizations as well as potential upgrades to take your site's speed to the next level.

Table based on the information in (Schaff, 2019)

| Criteria | Description | Tier |
|---|---|---|
| **Browser Caching / Expires Headers / Cache Static Content** | Essential for improving performance; prevents re-downloading assets on every page load. | Tier I (Deal-Breaker) |
| **Image Compression and Optimization / Page Size** | Critical for reducing download size and improving load times. | Tier I (Deal-Breaker) |
| **Time to First Byte (TTFB) / Page Caching** | Vital for improving server response times and reducing load on the server through caching. | Tier I (Deal-Breaker) |
| **Render-blocking Resources / Critical CSS / Async CSS & JS** | Newer best practice improves performance significantly when combined with other optimizations. | Tier II (Upgrade) |
| **JavaScript Execution Time / 3rd Party Scripts** | Manage JavaScript execution and limit 3rd party scripts to optimize performance. | Tier II (Upgrade) |
| **Unused CSS & JS (Code Coverage)** | Further optimization by removing unnecessary global CSS and JS to reduce bloat. | Tier II (Upgrade) |
| **Minification** | Important for reducing asset sizes and improving download times. | Tier II (Upgrade) |
| **CDN** | Easy win for improving load times and reducing server | Tier II (Upgrade) |

| | |
|---|---|
| | load by using a content delivery network. |

Table based on the information in (MOZ, 2021)

| Criteria | Description |
|---|---|
| **Enable Compression** | Use Gzip to reduce the size of CSS, HTML, and JavaScript files. Avoid gzip for images; compress them separately. |
| **Minify and Bundle Resources** | Minify CSS, JavaScript, and HTML by removing unnecessary characters and comments. Combine files to reduce the number of requests. |
| **Reduce Redirects** | Minimize redirects to decrease wait times for HTTP request-response cycles and speed up loading. |
| **Remove Render-Blocking Resources** | Avoid or minimize JavaScript and CSS that blocks page rendering to speed up load times. |
| **Leverage Browser Caching** | Set caching to store data so repeat visitors don't reload the entire page. Manage caching expiration using tools like YSlow. |
| **Improve Server Response Time** | Optimize server performance to achieve a response time under 200ms. Address issues like slow database queries and insufficient memory. |
| **Use a Content Distribution Network (CDN)** | Distribute content across multiple servers globally to enhance speed and reliability. |
| **Optimize Images** | Resize and compress images appropriately. Use the right format (PNG for graphics, JPEG for photos) and CSS sprites to reduce HTTP requests. |
| **HTTP/2** | Enable HTTP/2 to process multiple requests concurrently, improving page load speed. |
| **Preconnect / Prefetch / Preload** | Use techniques to pre-establish connections or fetch resources ahead of time to speed up loading. |
| **First Paint & First Contentful Paint** | Track when the first visual change or content appears on the screen to gauge initial load times. |
| **First Meaningful Paint** | Measure when the main content of the page becomes visible to the user. |
| **Time to Interactive** | Determine when the page is fully interactive and usable by the user. |
| **DOM Content Loaded** | Monitor when the HTML document has been completely loaded and parsed. |

The website (Pingdom tools, n.d.) can be used to test my website to see if there are any possible performance upgrades. Here are the mentioned techniques from the website.

| Criteria | Description |
|---|---|
| **Compress components with gzip** | Compress files like CSS, HTML, and JavaScript to reduce size. |
| **Add Expires headers** | Set expiration dates for cacheable resources to reduce load times. |
| **Use cookie-free domains** | Serve static content from domains that don't send cookies, reducing data overhead. |
| **Avoid URL redirects** | Minimize redirects to improve page load times. |

| Reduce DNS lookups | Reduce the number of domain name system queries to speed up the page load. |
| --- | --- |
| Avoid empty src or href | Ensure all src or href attributes are filled to prevent unnecessary requests. |
| Put JavaScript at the bottom | Move JavaScript to the bottom of the page to prevent it from blocking page rendering. |

Table based on the information on (Tips to improve website speed | How to speed up websites, n.d.) and (What is lazy loading?, n.d.).

| Criteria | Description |
| --- | --- |
| Optimize Images | Reduces image load time by compressing images and lowering their resolution and dimensions. This minimizes the size of the image files, improving overall page load speed. |
| Limit the Number of HTTP Requests | Fewer HTTP requests mean fewer round trips to the server, reducing load times. Minimizing the number of assets loaded (images, scripts, etc.) boosts the page's loading performance. |
| Use Browser HTTP Caching | Saves static files in a temporary cache, allowing returning visitors to load pages faster without fetching all assets again. This shortens load time for repeat visitors. |
| Remove Unnecessary Render-blocking JavaScript | Prevents the loading of non-essential code before the important page content, ensuring the main page elements load quicker, reducing initial load time. |
| Limit the Use of External Scripts | Reduces reliance on third-party scripts, which can delay page loading and cause layout shifting. By limiting external scripts, the page loads more smoothly and faster. |
| Limit Redirect Usage | Reducing the number of redirects prevents additional delays in reaching the final destination page, speeding up the time it takes for a user to access the desired content. |
| Minify CSS and JavaScript Files | Eliminating unnecessary characters and whitespace from code reduces file sizes, resulting in faster loading times and lower bandwidth usage. |
| Use Effective Third-party Services | Ensures that key functions such as hosting, DNS resolution, caching, and cybersecurity are handled efficiently. Fast hosting, DNS services, and CDN caching help deliver content quickly. |
| Lazy Loading | Defers loading of certain resources (like images) until needed, improving initial page load speed. Reduces load times for images |

| | below the fold, saving bandwidth and improving performance. |
|---|---|

With all this information it makes it easier to optimize my page loading speeds. By for example starting with the most appearing criteria such as image compression and caching and slowly working my way through the list of all the optimizations.

## Performance test

In this section I will test the performance of my website

# How many concurrent users should my website be able to handle?

## Competitive analysis

In the area of scalability there are 2 main factors: Hosting and Architecture. But before getting into that, how many users should my website be able to handle? Here is a comparison of different eCommerce websites and their average visits per day

| eCommerce Website | Average Visits per Day |
|---|---|
| Bol.com | 576.700 |
| Amazon.com | 27.400.000 |
| Aliexpress.com | 2.300.000 |
| Raphnet-tech | 143 |
| Azerty.com | 4.800 |
| Marktplaats.nl | 270.000 |

I was originally thinking to make something like raphnet-tech. But this would be no challenge in scalability. Therefore, I will challenge myself and choose something more difficult, like bol.com.

With the following formula from the source (Dehran, 2024). I can calculate the concurrent users bol.com handles. The values I input in the formula are found on (Bol.com Website Traffic, Ranking, Analytics [August 2024], 2024)

Concurrent Visitors = Per Day visits / Peak hours * (60/Average duration per visit in minutes)

Concurrent Visitors = 576,700 / 4 * (60 / 8) = 1,086,562.5

So roughly a million. I will try to build a website that can scale out to handle one million users at the same time.

# Which software architecture provides performance and efficient scalability up to one million concurrent users?

I am looking for an architecture which can scale out efficiently to handle one million concurrent users. With that in mind, it could help filter out architectures.

Best architecture practises according to (Perco, 2024). Marked in green are the pros which I look for in an architecture. Marked in red are the cons which are a dealbreaker for my project.

| Architecture Type | Pros | Cons |
|---|---|---|
| **Monolithic Ecommerce Architecture** | - **Simplicity**: Easy to develop, deploy, and manage.<br>- **Reliability**: Predictable with tight integration. | - **Scalability issues**: Hard to scale specific parts.<br>- **Rigidity**: Changes affect the entire system. |
| **Microservice Ecommerce Architecture** | - **Scalability**: Independent scaling of services.<br>- **Flexibility**: Easier to update and maintain specific parts. | - **Complexity**: Requires managing multiple services.<br>- **Resource-intensive**: Higher overhead. |
| **Two-Tier Ecommerce Architecture** | - **Performance**: Improves performance by distributing tasks.<br>- **Easily manageable**: Separation of client and server logic. | - **Limited scalability**: Can become a bottleneck as user base grows.<br>- **Security risks**: More exposure points. |
| **Three-Tier Ecommerce Architecture** | - **Scalability**: Each layer can be scaled independently.<br>- **Flexibility**: Easier to modify individual layers. | - **Complexity**: Harder to manage with more components.<br>- **Cost**: Potentially higher operational costs. |
| **Headless Ecommerce Architecture** | - **Flexibility**: Use any frontend for different devices.<br>- **Enhanced user experience**: Quick updates to frontend. | - **Complexity**: Requires advanced skills to manage.<br>- **Integration challenges**: Connecting multiple frontends to the backend. |
| **SaaS Ecommerce Architecture** | - **Cost-effective**: Low initial investment.<br>- **Maintenance-free**: Provider manages updates and security. | - **Limited control**: Dependent on third-party provider.<br>- **Customization limits**: Less flexible than other architectures. |

Here are the most common types of eCommerce architecture according to (BigCommerceTeam, Ecommerce Website architecture (Best Practices + Your Options), 2024).

| Architecture | Pros | Cons |
|---|---|---|
| Two-tier | - **Simplicity**: Easy to implement for smaller businesses.<br>- **Performance**: Allows fast | - **Limited scalability**: Struggles with handling larger data or complex logic. |

| | | |
|---|---|---|
| | client-server communication. | - **Performance**: Can degrade as demands grow. |
| Three-tier | - **Scalability**: Independent scaling of presentation, business, and data layers.<br>- **Flexibility**: Each layer can be developed and maintained separately. | - **Complexity**: More challenging to set up and maintain.<br>- **Cost**: Requires more infrastructure, leading to higher expenses. |
| SaaS | - **Quick setup**: Easy to launch for new businesses.<br>- **Low maintenance**: Provider handles hosting, updates, and security. | - **Limited control**: Customization and configuration options are limited.<br>- **Dependence**: Relies on the SaaS provider for updates and performance. |

With my preferences we can narrow down the decision to the following architectures:
**Microservice Architecture**, **Three-Tier Architecture**, **Headless Architecture**. Here is a closer look on those 3 architectures.

| Approach | Pros | Cons |
|---|---|---|
| **Headless E-commerce** | - **Flexibility and Agility**: Enables quick adaptation to customer expectations and market trends.<br>- **Seamless Omnichannel Experiences**: Delivers consistent shopping experiences across multiple channels.<br>- **Scalability and Performance**: Allows independent scaling of front-end and back-end systems, improving performance during high traffic. | - **Complex Implementation**: Requires integration of various technologies and can complicate the development process.<br>- **Dependency on APIs**: Relies heavily on APIs, which can lead to issues if not managed properly.<br>- **Increased Maintenance**: Requires ongoing management of multiple technologies and platforms. |
| **Microservices** | - **Scalability and Resilience**: Allows independent scaling based on demand, improving resource allocation.<br>- **Agility and Innovation**: Fosters an autonomous development approach, accelerating innovation and deployment.<br>- **Fault Isolation**: Issues in one service do not affect the entire system, enhancing reliability and maintenance ease. | - **Higher Complexity**: Managing multiple microservices can lead to increased architectural complexity. Inter-service<br>- **Communication Overhead:** Requires efficient communication between services, which can introduce latency.<br>- **Deployment Challenges**: Managing deployments of numerous independent services can be cumbersome. |

Information based off (Hasan, 2023).

| Architecture | Pros | Cons |
| --- | --- | --- |
| 3-Tier | **- Scalability:** Independent scaling of each tier.<br>**- Maintainability:** Better organization improves maintainability. | **- Scalability:** Centralized design can introduce bottlenecks.<br>**- Complexity:** Complexity in managing interactions between tiers. |
| Microservices | **- Scalability:** Highly scalable; individual services can be scaled independently.<br>**- Maintainability:** Loose coupling allows independent updates. | **- Complexity:** Increased complexity in managing numerous services.<br>**- Performance:** Communication overhead can introduce latency. |

Info based off (Tozzi, 2023)

| Criteria | Three-Tier Architecture | Microservices Architecture |
| --- | --- | --- |
| Modularity | More modular than monolithic architecture. | Highly modular, with each service focusing on a specific function. |
| Codebase Separation | Allows separation of application codebase into distinct parts. | Enables high separation between application parts, enhancing flexibility. |
| Deployment Independence | Components can be deployed independently, simplifying the deployment process. | Components can also be deployed independently, but requires more complex management. |
| Security Benefits | Separation reduces the impact of breaches on other components. | Offers security benefits, as issues in one microservice won't necessarily affect others. |
| Performance | Issues in one layer can affect overall app performance. | Performance is improved; issues in one service do not cause the entire app to fail. |
| Complexity | Simpler to build, deploy, and manage. | More complex to create and manage due to numerous individual components. |
| Scalability | Scalable, but less efficient than microservices for complex apps. | Highly scalable with granular scaling capabilities for individual components. |
| Deployment Environment | Suitable for applications deployed on one or a few servers. | Best for distributed environments that maximize scalability and resilience. |

| | | |
|---|---|---|
| **Development Processes** | Suitable for teams with simpler development operations. | Ideal for teams with advanced CI/CD pipelines capable of managing multiple services. |
| **IT Team Readiness** | Better fit for small or less experienced IT teams. | Requires a more experienced IT team to support the complexity of microservices. |

Information based off (BigCommerceTeam, Ecommerce Microservices vs. Monolith Models vs Headless Commerce, 2024)

| Architecture Type | Pros | Cons |
|---|---|---|
| **Microservices** | - **Independent Scaling**: Each service can scale individually, preventing back-end slowdowns from front-end traffic.<br>- **Customization Opportunities**: Businesses can select specific services tailored to their needs, enabling better personalization.<br>- **Rapid Implementation**: Decentralized teams can deploy updates faster, improving responsiveness to market changes.<br>- **Best-of-Breed Solutions**: Allows the use of specialized services from different providers rather than a single, all-in-one solution. | - **Requires Organizational Changes**: Moving to microservices may necessitate restructuring teams for better collaboration.<br>- **May Need Infrastructure Updates**: New tools may be required for managing microservices, adding complexity.<br>- **Potential High Costs**: Fully decoupled systems can be expensive due to multiple service providers. |
| **Headless Commerce** | - **Flexible Content Delivery**: Front-end and back-end can operate independently, enabling easier updates.<br>- **Agility in Customer Service**: Brands can quickly adapt to customer expectations and market trends.<br>- **Multiple Front-End Options**: Different user interfaces can be created while using a single back-end system, enhancing the customer experience. | - **Incremental Transition Challenges**: Transitioning can be complex, as it requires gradual separation of components.<br>- **Potential Complexity**: Managing multiple front-ends can introduce operational challenges.<br>- **API Dependency**: Reliance on APIs can lead to risks if any API issues occur. |

In conclusion, I have the option of choosing either a **Microservice Architecture, Three-Tier Architecture** or a **Headless Architecture**. While the microservice architecture excels in scalability, based off my research, all three options score similarly on the non-functional requirements I was considering. I could consider not choosing a **Three-Tier** because it is inefficient to scale, and efficient scalability is one of my main focusses in this project which would leave the decision between a **Microservice Architecture** or a **Headless Architecture**.

# How does my website's performance scale as the number of concurrent users increases?

Load test the website

# Which type of hosting provides high performance and automatic scaling?

## Best Practices

The type of hosting not only impacts the performance but also the scalability. With a good host you will be able to easily scale your application automatically. I compared hosting services while looking for eCommerce best practises.

Best hosting options according to (Newland, 2024)

| Hosting Option | Description |
|---|---|
| **Cloud Hosting** | Offers flexibility and scalability, allowing resources to be adjusted as needed. Ideal for sites with fluctuating traffic or rapid growth. Options include VPS, managed cloud, or PaaS. Cost-effective with a pay-as-you-go model. |
| **Dedicated Server** | Provides maximum control and performance, ensuring consistent operation for large B2B sites with complex requirements. A dedicated server is more expensive but guarantees high uptime and performance. |
| **Enterprise-grade Hosting** | Included with many B2B SaaS platforms, this hosting simplifies setup and reduces in-house technical needs. Scaling is automatic, and costs are predictable as part of a subscription fee. |

Best hosting options according (Low, 2024)

| Hosting Type | Description |
|---|---|
| **Shared Hosting** | Limited scaling options, usually better for small sites. |
| **VPS Hosting** | Easier to scale by adding more resources to your virtual server. |
| **Cloud Hosting** | Designed for scaling, allows you to add or remove resources quickly. |
| **Dedicated Hosting** | High scalability by upgrading hardware, best for large sites with heavy traffic. |

Hosting options according to this ecommerce blog (Samuelito, 2024)

| Hosting Type | Description |
|---|---|
| **Shared Hosting** | Most affordable option, but performance may suffer under high traffic, making it unsuitable for large eCommerce sites. |

| | |
|---|---|
| **VPS Hosting** | A good balance between cost and performance, suitable for growing businesses with moderate traffic needs. |
| **Dedicated Hosting** | <mark>Offers maximum performance</mark>, security, and control, ideal for large-scale operations but comes with a higher cost. |
| **Cloud Hosting** | <mark>Provides</mark> flexibility and <mark>scalability</mark>, allowing you to pay for only the resources used, making it great for fluctuating traffic. |

Based solely on performance, **Shared Hosting** isn't a viable option. In my experience, shared hosting does suffer in terms of performance. This leaves the choice to one of the following: **VPS Hosting**, **Dedicated Hosting** and **Cloud Hosting.**

Based on Scalability, All the hosting options are capable, but since most **cloud and enterprise-grade hosts are designed for auto-scaling**, they are the most practical choice. If, for some reason, I decide not to go with one of those, I would choose **VPS hosting** because it's easy to scale. While dedicated hosting is an option in theory, it would take a lot of time to set up manually, so it would be more of a last resort.

## Load Testing

<mark>Validate scalability here</mark>

# What are the best practices for enhancing and maintaining security in eCommerce platforms?

## Best Practices

To find the best practises for security I checked multiple websites and listed the best practises they mention.

Here are the best practises according to (Elitery, 2022)

| E-Commerce Security Tips | Description |
| --- | --- |
| **Use SSL and Follow PCI DSS Security Standards** | Secure Socket Layers (SSL) are essential for website authentication and data protection. Adhering to PCI DSS security standards enhances the security of financial transactions, helping validate credit card payments and preventing fraudulent transactions. |
| **Complete Website With DDoS and Firewall Application** | DDoS attacks can render online banking sites inaccessible. Utilizing reliable third-party applications like CloudFlare and Sucuri can mitigate DDoS attacks. Firewalls are also critical for protecting against SQL Injection and cross-site scripting (XSS) attacks. |
| **Always Update** | Many security incidents arise from outdated systems. Regular updates are necessary to safeguard your e-commerce site against vulnerabilities that intruders can exploit through old code. |
| **Have Layered Security** | Implementing multi-factor authentication, stronger passwords, address verification systems (AVS), and security warning systems can enhance e-commerce security. If your site integrates with external APIs, ensure verification and encryption measures are in place at each gateway. |
| **Selected Data Selection** | Not all data should be stored in the backend, especially sensitive information like customer credit card data. Compliance with PCI DSS certification requires that sensitive data not be stored on the transaction site, even if encrypted. |

Best practises according to (Derachits, 2024)

| Best Practice | Description |
| --- | --- |
| **Embrace Multi-Factor Authentication** | Implement multi-factor authentication methods like 2-step verification to enhance access security. |

| Use Stronger Passwords | Require strong passwords for accounts, incorporating special characters, numbers, and varying letter cases. |
|---|---|
| Keep Your E-commerce Website Up-to-Date | Regularly update software to patch vulnerabilities and maintain the latest security measures. |
| Use HTTPS Security | Switch from HTTP to HTTPS by using SSL to encrypt sensitive customer data during transactions. |
| Use a Firewall | Employ firewalls to monitor and block suspicious traffic while allowing authorized access to your site. |
| Only Store Necessary Customer Data | Limit the collection and storage of customer data to only what is essential for business operations. |
| Have a Secure E-commerce Platform | Choose an e-commerce platform that supports robust security measures and integrates well with security tools. |

The OWASP top 10 web application security risks (OWASP Top Ten | OWASP Foundation, n.d.). For more elaborate information about each risk, there are sub-pages for each of the risk explaining how I can tackle them.

| OWASP Category | Relevance to E-commerce | Actions to Mitigate |
|---|---|---|
| A01:2021 - Broken Access Control | Critical for preventing unauthorized access to user accounts, payment information, and admin functions. | Implement role-based access control (RBAC), conduct regular access reviews, and use secure coding practices. |
| A02:2021 - Cryptographic Failures | Essential for protecting sensitive data such as credit card details and personal information from exposure. | Use strong encryption standards (e.g., AES, RSA), employ secure protocols (e.g., TLS), and regularly audit cryptography practices. |
| A03:2021 - Injection | Vulnerabilities like SQL injection and XSS can compromise sensitive data and manipulate application behaviour. | Use prepared statements and parameterized queries, validate and sanitize user input, and implement Content Security Policy (CSP). |
| A04:2021 - Insecure Design | Flaws in design can introduce vulnerabilities; secure design practices help identify risks early. | Conduct threat modeling, apply secure design patterns, and involve security teams in the design phase. |
| A05:2021 - Security Misconfiguration | Misconfigurations during deployment can expose the application; regular audits are necessary. | Regularly review and test configurations, automate configuration management, and implement least privilege principles. |

| A06:2021 - Vulnerable and Outdated Components | Using outdated libraries can expose the application to known vulnerabilities; regular updates are critical. | Maintain an inventory of components, apply security patches promptly, and use tools for vulnerability scanning. |
|---|---|---|
| A07:2021 - Identification and Authentication Failures | Flaws can lead to account takeovers; robust authentication mechanisms are necessary. | Implement multi-factor authentication (MFA), enforce strong password policies, and use secure session management. |
| A08:2021 - Software and Data Integrity Failures | Ensuring the integrity of software updates and data is crucial to prevent manipulation and vulnerabilities. | Use checksums or digital signatures for updates, implement secure CI/CD practices, and regularly audit data integrity. |
| A09:2021 - Security Logging and Monitoring Failures | Effective logging is vital for detecting fraud and responding to security incidents. | Implement comprehensive logging of critical actions, utilize centralized log management solutions, and regularly review logs for anomalies. |
| A10:2021 - Server-Side Request Forgery | SSRF vulnerabilities can lead to unauthorized access to internal resources, making them a concern for e-commerce platforms. | Validate and sanitize user-supplied URLs, implement network segmentation, and monitor server-side requests. |

I plan to categorize security practices and risks using the MoSCoW method, prioritizing them from "must" to "could." This approach will help me implement the most critical measures first, followed by the less essential ones.

## Security Testing

Test the implemented security measures.

## Static program analysis

Set up static code analysis to detect security breaches.

# What are the best practices for ensuring maintainability in eCommerce platforms?

## Best Practices

Maintenance plays a crucial role in the success of any eCommerce platform. According to (BigCommerceTeam, Ecommerce Website Maintenance Generates Wins Now and for Later, 2024), here are some essential maintenance tasks that should be prioritized once the project is live

| Aspect | Description | Importance |
|---|---|---|
| Mitigating cyber threats | Regular updates to security software and plugins to protect customer data from hackers. | Reduces chances of cyber attacks and data breaches. |
| Maximizing website uptime | Ensuring the site remains live and accessible for customers. | Prevents costly downtime and loss of business. |
| Preventing data loss | Regular backups of databases and key systems. | Safeguards data in case of a catastrophe. |
| Maintaining compliance | Staying updated with legal standards like PCI-DSS and GDPR. | Ensures compliance with regulations, avoiding penalties. |
| Uptime consistency | Ensuring site availability for users at all times. | Crucial for seamless customer transactions and user experience. |
| Website security | Applying security patches and maintaining an SSL certificate. | Protects the site from emerging security threats. |
| Data backups | Taking regular snapshots of databases and key systems. | Provides insurance against data loss and improves recovery from cyber attacks or system failures. |
| Broken links | Running reports to find and fix broken links. | Improves site quality and customer experience. |
| Page speed | Optimizing load times to make the site faster. | Impacts conversion rates and customer satisfaction; slow load times reduce sales. |
| Updating outdated content | Regularly auditing and updating content to maintain accuracy. | Increases customer confidence and reflects business professionalism. |
| Priority keyword rankings (SEO) | Reviewing and retooling SEO to keep high search engine rankings. | Helps attract more customers through better search visibility. |
| Promotions and price changes | Regularly adjusting pricing to stay competitive. | Ensures the business remains competitive in an open market. |
| More prone to cyber attacks | Neglecting updates makes ecommerce sites attractive targets for hackers. | Increases the risk of data breaches and theft of sensitive information. |

| Slow website load times | Poorly maintained sites have slower load times. | Conversion rates drop significantly with every extra second of load time. |
|---|---|---|
| Higher costs due to recurring bugs | Neglected websites are more prone to bugs and crashes. | Results in higher costs for fixing issues and lost opportunities during downtime. |
| Loss in customer trust | Regular site issues erode customer confidence. | Customers are less likely to buy from a site they don't trust, especially with personal and financial information. |
| Outdated and misleading content | Inaccurate information discourages purchases and leads to customer dissatisfaction. | Regular content audits maintain accuracy and customer satisfaction, especially with new product launches. |

Best practices according to (Rawat, 2024)

| Best Practice | Description |
|---|---|
| Understanding Your Website's Needs | Regularly assess your website for outdated content, security vulnerabilities, and performance issues. |
| Setting Concrete Objectives | Define specific goals for your website maintenance, such as boosting sales, enhancing user experience, or improving search rankings. |
| Prioritizing Critical Components | Focus on the most important aspects of the website, including backups, security, and user experience, to save time and resources. |
| Creating a Consistent Maintenance Schedule | Establish a regular routine for updates, security checks, and backups to ensure consistent performance. |
| Delegating Tasks Effectively | Assign maintenance responsibilities to team members or specialists to ensure tasks are handled efficiently. |
| Documenting Procedures for Clarity | Create step-by-step documentation for maintenance tasks to maintain clarity, consistency, and ease of training new team members. |
| Budgeting Wisely for Tools and Resources | Allocate a budget for necessary tools, plugins, or services that support website maintenance and deliver a strong return on investment. |
| Ensuring Robust Security Measures | Implement security protocols such as SSL encryption, regular audits, and timely updates to protect customer data and enhance credibility. |

After reviewing both sources, I noticed that the non-functional requirement maintenance involves various other nonfunctional requirements like performance, security, and usability. Many of these practices have already been covered in their own sections of the research, and most are primarily relevant once the project is live.

# What are the best practices for improving usability in eCommerce platforms?

## Best Practices

Usability plays a major factor in conversion rate. Therefore, it's important to make the website as attractive and user friendly as possible. Here are some best practises on the field of usability according to (Markovich, 2024)

| Best Practice | Description |
| --- | --- |
| Attractive Product Display | Display products in a visually consistent and attractive way. |
| Personalize Suggestions | Tailor product suggestions and deals to individual customers. |
| Implement clear navigation | Make it easy for customers to find what they're looking for with intuitive navigation. |
| High-quality images | Use professional, high-quality images that capture attention. |
| Insert enticing calls to action | Include clear and persuasive calls to action to encourage customers to make a purchase or explore more. ("Shop Now", "Buy Now", "Add to Cart", "Learn More" etc.) |

**"Personalize Suggestions"** is an interesting best practice because it needs to be considered before I begin the project, unlike the other practices that can be implemented at a later stage.

Here are most best practices according to (Cooper, 2023)

| Best Practices for Optimizing Website Usability | Details |
| --- | --- |
| Ease Navigation | User-friendly navigation is essential for driving conversions. Ensure visitors can easily find what they need. Consider implementing a sitewide search function and navigation tools that are easy to locate. Use a simple navigation bar with practical categories. The checkout button should be displayed on all pages. |
| Streamlined Checkout | Lengthy checkout processes can lead to cart abandonment. Minimize the required information from users, such as using a checkbox to autofill shipping details. Allow guest checkouts to simplify the process and provide an option to save credit card information for returning customers. Carewell's case study shows a direct correlation between stored payment options and a **200% increase in conversion rates**. |
| Search Engine Optimization | Optimize site search functionality to enhance product findability. Most users expect a search box in the top right corner. Use pre- |

| | filling based on popular searches and keep search input visible during results display. Implementing effective SEO strategies can also drive relevant traffic to your site, thereby increasing conversion rates. |
|---|---|
| **Optimized Page Load Speeds** | Fast loading times are critical for retaining customers. Ensure images are appropriately sized (under 1000 pixels) and consider compressing them to improve load speed. Reduce unnecessary redirects as they slow down the website. |
| **Optimize for Mobile use** | A mobile-friendly interface is essential for modern e-commerce. Ensure compatibility with mobile technologies and optimize for mobile usability by increasing button sizes, simplifying navigation, and offering features like credit card scanning. Maintain a single domain for both mobile and desktop versions to avoid confusion. Successful mobile optimization has led to significant increases in conversion rates for companies like Ice Jewellery. |
| **Accessible for all Users** | Accessibility is key for a usable e-commerce experience. Implement features like voice search, keyboard navigation, and alt-tags for images. Use high-contrast color schemes and provide text alternatives for media. Testing accessibility through crowdsource testing can identify challenges and improve user experience. Accessibility upgrades can be low-cost and enhance overall usability, thus increasing conversions. |
| **Design your pages better** | Ensure a clear and straightforward pathway to products. Differentiate information between the homepage and product pages. Highlight product information effectively and organize content clearly. Including customer ratings and reviews can help inform potential buyers, increasing the likelihood of purchase. Skullcandy's award-winning design focuses on clarity and user engagement. |
| **Focus on A/B Testing** | A/B testing allows you to identify usability issues through randomized experiments comparing two variants. This cost-effective method can help you understand user reactions to changes in design or functionality, thus minimizing the risk of negative impacts on user experience. Insights from A/B testing and other usability testing methods can inform improvements aligned with consumer needs. |

# Usability testing

Validate that my website is understandable and user friendly.

# Postface

So, what are the best practices in eCommerce. Now that I answered my sub-questions, I have a better idea of what I should do next.

## Page Loading Speeds

For page loading speeds there are many tips in its own section. I would recommend going through them and prioritize them using the MoSCoW method. Something noticeable from this section of the research is that I want my pages to load within 1 second (with a maximum of 2 seconds if it really doesn't work out). This is based on research on conversion rates. See the dedicated section on page loading speeds for more details.

Page Loading Speeds: **1 second** (worst case scenario max 2 seconds)

## Concurrent Users

I've decided to build a website with the scalability of Bol.com. Based on my calculations, this means the website needs to be capable of handling 1 million concurrent users.

Concurrent Users: **One Million**

## Hosting

For hosting, the three hosts left were: **Cloud Hosting**, **Enterprise-Grade Hosting** and **VPS Hosting**. I decided that it would be more convenient to choose **Cloud** or **Enterprise-Grade** hosting because they usually have auto-scaling built in which is amazing in my case. Therefore VPS hosting is more of a last resort.

Hosting Choice: **Cloud Hosting** or **Enterprise-Grade Hosting** (VPS hosting if both don't work out)

## Architecture

My choice for the architecture should be one of the following: **Microservice Architecture**, **Three-Tier Architecture**, **Headless Architecture.** In short, scaling a **Three-Tier** is not as convenient as scaling a microservice architecture, but it's simpler to develop and manage. The headless architecture is completely different than these two architectures, and shines in its omnichannel possibility. However, I don't need that, but that doesn't make it a bad choice.

Meaning the decision is between **Microservice Architecture** and **Headless Architecture**. I am choosing for the Microservice Architecture for the sake of that I want to learn better how to implement it.

Architecture Choice: **Microservices** (But Headless is also a very good choice)

## Security

For security I collected many tips I could work on later after the project has a walking skeleton. Something to keep in mind in the project starting phase is to pick a host with many built in security functionalities so that I don't have to implement them myself (Think of **DDoS protection** for example). And for example, for authentication, choose a technology that allows **Multi Factor Authentication**. More tips in its own dedicated section

## Usability

For usability I mainly found tips to enhance the user experience, but I can also derive design choices from the tips I found.

Important Usability Design Choices: **Show Personalised Products**, **Optimize Search Engine**, **Mobile and PC interface**.

## Bibliography

BigCommerceTeam. (2024, September 13). *Ecommerce Microservices vs. Monolith Models vs Headless Commerce*. From BigCommerce: https://www.scribbr.nl/bronvermelding/generator/mappen/1hjpTmj2KusEWXGfSipOq5/lijsten/6Fadug4ps5uSvtc2JBLjuK/bronnen/O8eSNUr5q9CBsgCPOqMCN/bewerk/

BigCommerceTeam. (2024, September 5). *Ecommerce Website architecture (Best Practices + Your Options)*. From BigCommerce: https://www.bigcommerce.com/articles/ecommerce-website-development/ecommerce-architecture/

BigCommerceTeam. (2024, January 23). *Ecommerce Website Maintenance Generates Wins Now and for Later*. From BigCommerce: https://www.bigcommerce.com/articles/ecommerce/website-maintenance/

*Bol.com Website Traffic, Ranking, Analytics [August 2024]*. (2024, September 28). From Semrush: https://www.semrush.com/website/bol.com/overview/

Cooper, M. (2023, August 17). *8 Best Practices for ecommerce usability | BigCommerce*. From BigCommerce: https://www.bigcommerce.com/blog/ecommerce-usability/

Dehran, B. (2024, January 17). *How to measure the concurrent users that an eCommerce can handle?* From Cloudkul: https://cloudkul.com/blog/what-is-concurrent-users/

Derachits, V. (2024, August 5). *7 E-commerce security Best practices*. From Amasty: https://amasty.com/blog/7-e-commerce-security-best-practices/

Elitery. (2022, April 6). *E-Commerce Security Best Practices Tips*. From Elitery: https://elitery.com/articles/e-commerce-security-best-practices-tips/

Hasan, S. (2023, December 25). *Microservices vs. 3-Tier vs. Monolithic*. From Medium: https://saedhasan.medium.com/microservices-vs-3-tier-vs-monolithic-9073c80df468

*Headless e-commerce vs. traditional e-commerce*. (n.d.). From Sitecore: https://www.sitecore.com/explore/topics/headless-ecommerce/traditional-ecommerce-vs-headless-ecommerce

Low, J. (2024, July 31). *Best Scalable Hosting Solutions for High Traffic Websites*. From HostScore: https://hostscore.net/choose/best-scalable-hosting-solutions/

Markovich, P. (2024, June 21). *Ecommerce UX Best Practices to Boost Conversions in 2024 — Alva Commerce*. From Alva Commerce: https://alvacommerce.com/e-commerce-ux-best-practices-in-2024/

MOZ. (2021, May 24). *Page Speed*. From MOZ: https://moz.com/learn/seo/page-speed

Newland, W. (2024, June 20). *Enterprise-Grade Web Hosting Explained*. From SoBold: https://www.djust.io/blog-posts/ecommerce-scalability-and-performance

*OWASP Top Ten | OWASP Foundation*. (n.d.). From OWASP: https://owasp.org/www-project-top-ten/

Perco, A. (2024, June 26). *Ecommerce Architecture: Definition, Importance, & Best Practices*. From Semrush Blog: https://www.semrush.com/blog/ecommerce-architecture/

*Pingdom tools*. (n.d.). From https://tools.pingdom.com/

Rawat, R. (2024, March 18). *Ecommerce Website Maintenance: The Ultimate Guide*. From Narola Infotech: https://www.narolainfotech.com/blogs/ecommerce-website-maintenance/

Samuelito. (2024, July 4). *17 Tips to Optimize your ecommerce Hosting for maximum performance*. From Saucal: https://saucal.com/blog/ecommerce-hosting/

Schaff, A. (2019, August 6). *The Page Speed Optimization Hierarchy*. From Portent: https://www.portent.com/blog/design-dev/page-speed-optimization-hierarchy.htm

Technologies, C. (2021, August 3). *E-commerce Website Functional and Non-Functional Requirements With List & Examples*. From nasscom community: https://community.nasscom.in/communities/mobile-web-development/e-commerce-website-functional-and-non-functional-requirements

*Tips to improve website speed | How to speed up websites*. (n.d.). From Cloudflare: https://www.cloudflare.com/learning/performance/speed-up-a-website/

Tozzi, C. (2023, September 11). *Three-tier vs. microservices architecture: How to choose*. From App Architecture: https://www.techtarget.com/searchapparchitecture/tip/Three-tier-vs-microservices-architecture-How-to-choose

*What is lazy loading?* (n.d.). From Cloudflare: https://www.cloudflare.com/learning/performance/what-is-lazy-loading/

Wiegand, M. (2022, April 20). *Site Speed is (Still) Impacting Your Conversion Rate*. From Portent: https://www.portent.com/blog/analytics/research-site-speed-hurting-everyones-revenue.htm