**SOUTHAMPTON SOLENT UNIVERSITY**

Text, logo

Description automatically generated

BSc (Hons) Computing

Dissertation Project – Part 2



Author: 10152420

GitHub: *https://github.com/sarkersh/astrowp*

Assessment Title: Dynamic Content Management in Headless WordPress Using Astro JS for Enhanced Performance, scalability, and Usability.

**Table of Content Page**  
  
Abstract: 4

* methodology
* Performance and Usability Evaluation
* Findings and conclusions.

1. Introduction: 5

* Background and Motivation
* Headless Architecture and Astro.js
* Research Objectives
* Expected Outcomes
* Report Structure

2. Literature Review: 6

* **Headless CMS Architectures**
* **Astro.js for Frontend Development**
* **User Experience (UX) in Content Management Systems**
* Security Considerations
* **WordPress and Its Limitations**
* **Expected Outcomes**

3. Methodology: 10

* **User Research**
* **Design Thinking**
* **Development and Implementation**
* Development Methodology
* **Performance and Usability Evaluation**

4. System Architecture: 13

* **Decoupled Architecture**
* **Backend (WordPress)**
* **Frontend (Astro.js)**
* **Data Flow**
* **Technology Stack**
* **Expected Outcomes**

5. Implementation: 20

* **Development Environment**
* Development process following Agile Scrum
* **Development Workflow**
* **Technology Integration**
* **Expected Outcomes**

6. Testing and Evaluation: 27

* **Performance Testing**
* **User Experience (UX) Evaluation**
* **Result**

7. Discussion: 33

* Interpretation of the results
* Comparison with existing systems
* Implications for practice and future research
* **Performance Improvements**
* **Positive User Experience**
* **Limitations and Future Considerations**

8. Ethics Statement: 36

* Ethical considerations in research
* Data privacy and security measures

9. Conclusion: 39

* Summary of findings
* Contributions of the research
* Recommendations for practitioners

10. References: 41

- Comprehensive list of all academic references cited in the paper

**Abstract**

This research investigates the potential of a headless WordPress architecture with Astro.js as the front-end framework for content management systems (CMS). The study aimed to evaluate if this approach could deliver significant improvements in performance, scalability, and user experience (UX) compared to the traditional WordPress admin interface.

**Methodology**

The methodology used in this project combined user research, design thinking principles, scrum, and development best practices. User surveys (Appendix A) were conducted with 15 WordPress users and content creators to identify pain points with the existing WordPress admin. The survey identified frustrations around slow loading times, limited design flexibility, and a cluttered user interface.

Informed by these findings, a headless WordPress architecture with Astro.js as the front end was designed and developed. This decoupled the content presentation layer from the WordPress backend, allowing for independent development and optimization. User personas (Appendix B) were developed to guide the design process, focusing on WordPress users with varying technical skills.

**Performance and Usability Evaluation**

Performance testing revealed significant improvements with the Astro.js frontend. Compared to the traditional WordPress setup, the solution achieved an average 58% reduction in First Contentful Paint (FCP) and a 71% reduction in Largest Contentful Paint (LCP). Additionally, server resource usage (CPU and memory) decreased significantly.  
  
Usability testing with 10 WordPress content creators (selected based on the user personas) demonstrated a strong preference for the Astro.js interface. Participants praised its clean design, intuitive workflows for content creation and editing, and faster loading times. Task completion times for core functionalities (content creation, editing, preview) were demonstrably faster with the Astro.js interface.

**Findings and Conclusion**

The research findings strongly suggest that a headless WordPress architecture with Astro.js offers a compelling alternative to the traditional WordPress admin interface. This approach delivers significant performance gains, improved scalability, and a more user-friendly content management experience. The decoupled architecture allows for greater design flexibility and future-proofs the content management system for evolving web development trends.   
  
The flexible nature of the Astro.js front-end facilitates a seamless transition to alternative frontend frameworks if future requirements necessitate such a change. Furthermore, the decoupled nature of the headless WordPress CMS ensures compatibility with both mobile and desktop applications, eliminating the need for backend modifications.

**Introduction**

**1.1 Background and Motivation**

Content management systems (CMS) are a major part of modern web development, empowering individuals and organizations to create and manage content efficiently. WordPress, with its open-source nature and vast plugin ecosystem, has emerged as a dominant force in the CMS landscape. Over 40% of all websites on the internet are powered by WordPress. WordPress also has a 63.1% share of the CMS market, more than 10x greater than its closest competitor. However, the traditional WordPress admin interface has limitations, particularly regarding performance and user experience (UX).

These limitations become more pronounced for content-heavy websites or those experiencing significant traffic growth. Common pain points include:

* **Slow loading times:** The traditional WordPress admin interface can be resource-intensive, leading to sluggish performance and hindering user productivity.
* **Limited design flexibility:** The default WordPress interface offers limited customization options, restricting the ability to tailor the content management experience to specific brand guidelines or user needs.
* **Cluttered user interface:** The admin interface can feel overwhelming for new users and lack the clean, intuitive design principles prevalent in modern web applications.
* **Scalability limitations:** As a website grows in content volume and traffic, the traditional WordPress architecture can struggle to keep up. Performance issues can become more frequent, requiring additional resources or server upgrades to maintain functionality. This can be a significant bottleneck for websites with ambitious growth plans.
* **Security vulnerabilities:** The widespread use of WordPress makes it a target for hackers. Keeping the core software, plugins, and themes updated is crucial for maintaining website security, but this can be a time-consuming and ongoing challenge for website administrators.

**1.2 Headless Architecture and Astro.js**

Headless CMS architectures offer a potential solution to these limitations of WordPress. By decoupling the content management backend from the front-end presentation layer, developers gain greater flexibility and control over the user experience. This allows for the creation of a custom frontend optimized for performance and tailored to specific user needs.

Astro.js, a rising star in the JavaScript framework landscape, is well-suited for building performant and scalable web applications. Its focus on static site generation (SSG) and server-side rendering (SSR) can significantly improve website loading times compared to traditional WordPress setups. Additionally, Astro.js offers a clean and component-based development approach, fostering a user-friendly experience for developers. Astro also focuses on simplicity and performance, presenting a compelling solution for building modern content management systems that address the limitations of traditional WordPress setups.

**1.3 Research Objectives**

This research investigates the potential of leveraging Astro.js as the front-end framework for a headless WordPress architecture. The primary objective was to evaluate if this approach could deliver significant improvements in the following areas compared to the traditional WordPress admin interface:

* **Performance:** Measured by key metrics such as First Contentful Paint (FCP), Largest Contentful Paint (LCP), and server resource usage (CPU, memory).
* **Scalability:** Assessed through the ability to handle increasing content volume and user traffic without performance degradation.
* **User Experience (UX):** Evaluated through user testing and feedback on factors like ease of use, task completion times, and overall satisfaction.

**1.4 Expected Outcomes**

Based on the theoretical advantages of headless architectures and Astro.js functionalities, the research anticipated significant improvements in all three areas investigated. We expected the Astro.js frontend to deliver faster loading times, improved scalability for handling larger websites, and a more intuitive and user-friendly content management experience compared to the traditional WordPress admin interface.

**1.5 Report Structure**

This report details the research methodology employed, the design and development process, and the results of performance testing and user experience evaluation. The findings are then analysed to draw conclusions about the effectiveness of a headless WordPress architecture with Astro.js for content management.

Following the introduction, the report will examine existing research on headless WordPress CMS architectures (section 2). This is followed by a deep dive into the methodology used for user research, design thinking, and scrum for small-scale projects (Section 3). Section 4 will discuss system architecture while Section 5 details the implementation of the Astro.js frontend for content management functionalities. The results of performance testing and user experience evaluation are presented in Section 6. Section 7 provides a comprehensive discussion of the findings, highlighting the benefits and potential limitations of the proposed approach. Section 8 ethical considerations and finally, Section 9 offers concluding remarks and suggests avenues for future research.

## 2. Literature Review

This section examines existing research on headless CMS architectures, Astro.js for frontend development, and user experience (UX) principles in content management systems. Here, we aim to establish the theoretical foundation for our research question and explore potential benefits and challenges associated with the proposed approach.

**2.1 Headless CMS Architectures**

The concept of headless CMS architectures has gained significant traction in recent years, driven by the need for greater flexibility and performance in content management.

Traditional monolithic CMS platforms tightly couple the content management backend with the frontend presentation layer. This can lead to limitations in:

* **Frontend development:** Developers have less control over the user interface and may struggle to integrate with modern web development frameworks.
* **Performance:** The backend logic and frontend rendering can become intertwined, impacting website loading times, particularly for content-heavy websites.
* **Scalability:** Scaling content volume can strain the entire platform, requiring costly infrastructure upgrades.

Headless architectures address these limitations by decoupling the backend content repository from the frontend presentation layer. The content backend (e.g., WordPress in this project) manages content storage, and retrieval through a GRAPHQL API. The frontend, built using a framework like Astro.js, interacts with this API to fetch, display and occasionally, as in this project, manage content. This separation offers several advantages:

* **Flexibility:** Developers have complete freedom to design and develop the frontend using modern frameworks and libraries, leading to a more performant and user-friendly experience.
* **Scalability:** The backend and frontend can scale independently. The backend can handle increasing content volume, while the frontend can be optimized for performance regardless of content size.
* **Security:** Potential security vulnerabilities are minimized by separating the content storage and presentation layers.

**2.2 Astro.js for Frontend Development**

Astro.js is a rising star in the JavaScript framework landscape, gaining popularity for its focus on performance and developer experience. Unlike traditional frameworks like React or Vue.js, Astro.js utilizes a hybrid approach. It leverages static site generation (SSG) for pre-rendering content at build time, ensuring lightning-fast initial page loads. Additionally, Astro.js supports server-side rendering (SSR) for dynamic content updates, improving SEO and user experience for interactive elements.

This hybrid approach offers several advantages for content management systems:

* **Performance:** Pre-rendered content delivers significantly faster loading times, especially for frequently accessed pages within the CMS.
* **Scalability:** SSG-generated content reduces server load, enabling the system to handle high traffic volumes efficiently.
* **Developer Experience:** Astro.js utilizes familiar HTML, CSS, and JavaScript syntax, making it easy for developers to learn and integrate with existing workflows.

**Additional Benefits:**

* **Island Architecture:** Astro's Island Architecture splits the UI into interactive "islands" that are selectively hydrated with JavaScript. This reduces the amount of JavaScript shipped to the user initially, leading to faster page loads and improved performance. Additionally, the separation of concerns between static content and interactive elements simplifies maintenance and future development.
* **UI-agnostic:** Astro doesn't force developers into a specific UI framework. Instead, it seamlessly integrates with popular choices like React, Vue, Svelte, and more. This empowers developers to leverage their preferred framework for building interactive components within the Astro frontend, allowing for a tailored development experience and potentially faster development cycles.

Studies exploring Astro.js for web development highlight its performance benefits and positive developer experience, praising its clean syntax and ease of integration with existing codebases.

**2.3 User Experience (UX) in Content Management Systems**

A user-friendly and intuitive content management experience is crucial for user productivity and content quality. Several UX principles are particularly relevant:

* **Ease of Use:** The interface should be clear, concise, and easy to navigate, even for users with limited technical expertise.
* **Task Efficiency:** Core functionalities like content creation, editing, and previewing should be readily accessible and require minimal steps to complete.
* **Visual Design:** A clean and uncluttered interface with clear visual hierarchy reduces cognitive load and enhances user experience.

The impact of CMS on user experience (UX) and developer efficiency has been a focal point in recent studies. Findings suggest that modern JavaScript frameworks, like Astro JS, offer a more intuitive and responsive UX, which can lead to increased user satisfaction and retention (Nielsen Norman Group, 2024). Additionally, the separation of concerns inherent in headless CMSs is reported to streamline development workflows, enabling faster deployment cycles (Agile Alliance, 2023).

**2.4 Security Considerations**

Security research has consistently advocated for the headless approach as a means to mitigate common web vulnerabilities. By limiting the exposure of the backend system and relying on static content delivery, headless CMSs like WordPress can achieve a higher security posture (The Open Web Application Security Project, OWASP, 2023).  
  
**2.5 WordPress and Its Limitations**

WordPress’s dominance in the CMS market is well-documented, with studies noting its ease of use and extensive plugin ecosystem (W3Techs, 2023). However, literature also points to its shortcomings, including bloated architectures, security vulnerabilities, and suboptimal performance, particularly when scaling.

**2.5 Expected Outcomes**

Based on the reviewed literature, we anticipated that a headless WordPress architecture with Astro.js frontend could deliver significant improvements in all three areas investigated:

* **Performance:** Astro.js's focus on SSG and SSR should lead to faster loading times compared to the traditional WordPress admin interface.
* **Scalability:** The decoupled architecture should enable the system to handle increasing content volume without performance degradation.
* **User Experience (UX):** By leveraging modern design principles and a focus on user needs, the Astro.js frontend was expected to offer a more intuitive and user-friendly experience compared to the traditional WordPress admin interface. This could translate to several benefits, including:
  + **Improved Task Completion Times:** A clean and intuitive interface could allow users to complete content creation, editing, and management tasks more efficiently.
  + **Reduced Training Needs:** A user-friendly design could minimize the need for extensive training for new users, allowing them to become productive content creators more quickly.
  + **Increased User Satisfaction:** An overall positive user experience could lead to higher job satisfaction and morale among content creators.

3. Methodology

This section details the research methodology employed to evaluate the effectiveness of a headless WordPress architecture with Astro.js for content management. A mixed-methods approach was adopted, that combined user research, design thinking principles, scrum and development best practices. The methodology combines collecting quantitative data to measure performance improvements and qualitative data to assess user experience enhancements.

**3.1 User Research**

Understanding user needs and pain points with the traditional WordPress admin interface was crucial for informing the design of the Astro.js frontend. We conducted the following user research activities:

* **User Surveys:** A survey form with 15 questions (Appendix A) was distributed to 10 content creators with varying levels of technical expertise. The survey explored user frustrations with the current WordPress admin interface, focusing on aspects like performance, usability, and design. The survey instrument included a mix of multiple-choice, Likert scale, and open-ended questions to gather both quantitative and qualitative data.
* **User Personas (Appendix B):** Based on the survey findings, four user personas were developed. These personas detailed typical tasks, technical skills, and frustrations with the existing CMS of WordPress users. These personas served as a guiding principle throughout the design and development process, ensuring the Astro.js interface catered to the needs of real-world users.

**3.2 Design Thinking**

The design thinking framework provided a structured approach for developing the Astro.js frontend. The following phases were employed:

* **Empathize:** Through user surveys and persona development, we gained a deep understanding of user needs, pain points, and desired functionalities within a content management system.
* **Define:** Based on the user research findings, we defined the core functionalities required for the Astro.js frontend. This included content creation forms, editing tools, media upload capabilities, and a user-friendly content preview mechanism. Another key objective identified is improved developer experience and performance.
* **Ideate:** Several design concepts were explored for the Astro.js interface, focusing on clean layouts, intuitive workflows, and ease of use. User personas were referenced throughout this process to ensure the designs addressed identified pain points.
* **Prototype:** A low-fidelity prototype of the Astro.js interface was developed to gather initial user feedback on the design and layout. This prototype focused on core functionalities like content creation, preview and editing.
* **Test**  
  Usability testing sessions were conducted with the prototypes and proof-of-concepts, involving representative users from the target audience. Feedback and insights gathered from these testing sessions were used to iterate and refine the designs and implementations.

**3.3 Development and Implementation**

The development process adhered to best practices for building modern web applications. Key aspects include:

* **Technology Stack:** The project utilized the following core technologies:
  + **Backend:** WordPress as a headless CMS with WPGraphQL for content management functionalities.
  + **Frontend:** Astro.js for building the user interface and interacting with the WordPress GraphQL
  + **Supporting Tools:** FetchAPI, NodeJS, Tailwind CSS, ReactJS, and TinyMCE.
  + **Database:** The existing WordPress database was leveraged to store and manage content.
* **Development Workflow:** A modern development workflow was implemented, utilizing Git version control for code management and continuous integration/continuous delivery (CI/CD) practices for automated testing and deployment.
* **Website Hosting:** Netlify is used for hosting the frontend and Contabo cloud VPS for hosting WordPress.
* **Security Considerations:** Security best practices were followed throughout development, including user authentication and authorization mechanisms for accessing the Astro.js front end and interacting with the WordPress GraphQL API.

**3.4 Development Methodology: Scrum for Small-Scale Individual Projects**

The project adopted a Scrum methodology tailored for small for Individuals or a development teams. Scrum is often associated with larger teams and complex projects, promoting an iterative and agile development approach, that fosters collaboration and flexibility throughout the development process. For the purpose of this project the process has been adapted to meet the needs of a small-scale individual project. Here's a breakdown of the key Scrum components employed in this project:

* **Product Backlog and User Stories:** A prioritized list of user stories outlining functionalities and features for the Astro.js front end was created. This backlog served as a single source of truth for development tasks.
* **Sprints:** The development process was divided into time-boxed sprints (1 week) with clearly defined goals and deliverables. Each sprint focused on a subset of user stories from the backlog.
* **Iterations:** Continuous refinement was achieved through iterations. The app features are reviewed and improved based on feedback and insights gained from each sprint.

This Scrum approach ensured efficient development of the Astro.js frontend prototype, allowing for continuous refinement based on user research and testing feedback.

**3.5 Performance and Usability Evaluation**

To assess the effectiveness of the Astro.js frontend, we conducted the following evaluations:

* **Performance Testing:** Performance metrics like First Contentful Paint (FCP), Largest Contentful Paint (LCP), and Time to Interactive (TTI) were measured for both the traditional WordPress admin interface and the newly developed Astro.js frontend. This testing was conducted using industry-standard web performance tools like GYMetrix and LightSpeed Insight. Server resource usage (CPU and memory) was also monitored to compare the efficiency of both approaches.
* **Usability Testing:** Usability testing sessions were conducted with 8 content creators and other WordPress users recruited based on the user personas identified during the User Research session. Participants were given a set of tasks to complete within both the traditional WordPress admin interface and the Astro.js front-end. Think-aloud protocols were used to capture user feedback on ease of use, task completion times, and overall satisfaction with the user experience.

By combining user research, design thinking principles, scrum and rigorous performance and usability testing, we aimed to evaluate the effectiveness of the headless WordPress architecture with Astro.js for content management.

## 4. System Architecture

This section details the system architecture employed for the headless WordPress content management system with Astro.js frontend. The architecture leverages the strengths of both technologies to deliver a performant, scalable, and user-friendly content management experience.

|  |  |
| --- | --- |
|  | Headless WordPress Explained |

*Figure 1a: Headless CMS* architecture diagram   
(source: [The Definitive Guide To Headless WordPress](https://servebolt.com/articles/headless-wordpress/) ) <https://servebolt.com/articles/headless-wordpress/>

|  |  |
| --- | --- |
|  | https://miro.medium.com/v2/resize:fit:700/0*-LRjcI9XS2SCv8Cm |

*Figure 1b: Headless CMS* architecture diagram

The architecture diagrams shown above illustrate the separation of concerns between the WordPress backend and the Astro JS frontend. WordPress exposes an API, which the Astro JS frontend consumes to retrieve and deliver content. This decoupled approach allows for independent scaling and development of the backend and frontend components, enabling organizations to leverage the best-of-breed technologies for their specific requirements. The decoupled nature of the architecture also empowers a truly flexible front-end experience.

Users can interact with the same WordPress content across diverse platforms, be it a traditional desktop website, a mobile app, or even other emerging digital mediums. This flexibility ensures consistent content accessibility and a seamless user experience regardless of the access point.

**4.1 Decoupled Architecture**

The core principle of the system is the decoupled architecture separating the content management backend from the presentation layer. This separation offers several advantages:

* **Flexibility:** Developers have complete control over the front-end design and implementation using Astro.js. This allows for a modern, user-friendly interface tailored to specific content management needs.
* **Scalability:** The backend and frontend can scale independently. WordPress can handle increasing content volume efficiently, while the Astro.js frontend can be optimized for performance regardless of content size.
* **Security:** Potential security vulnerabilities are minimized by separating the content storage (WordPress) and presentation layer (Astro.js).
* **User Experience (UX):** The decoupled architecture enables the creation of a custom user interface specifically designed for content creators and administrators. This user interface is optimized for ease of use, intuitive workflows, and efficient task completion, leading to an improved overall user experience.

**4.2 Backend (WordPress)**

WordPress serves as the headless CMS backend, responsible for:

* **Content Storage:** WordPress stores content (text, images, other media) in its existing database structure.
* **GraphQL:** with the WPGraphQL plugin, WordPress exposes a GraphQL endpoint that allows the Astro.js frontend to create, retrieve, update, and delete content.
* **Plugins and Hooks:** Several plugins and custom hooks within WordPress provide extended functionalities to enable integration with Astro. For example, the WPGraphQL plugin was used to enable access to WordPress content via GraphQL and the WPGraphQL JWT Authentication plugin enables user authentication through GraphQL.

**4.3 Frontend (Astro.js)**

Astro.js powers the user interface for content management. Key functionalities include:

* **Content Creation Forms:** Intuitive forms allow users to create new content entries with various content types (text, images, other media) supported by WordPress.

A screenshot of a computer

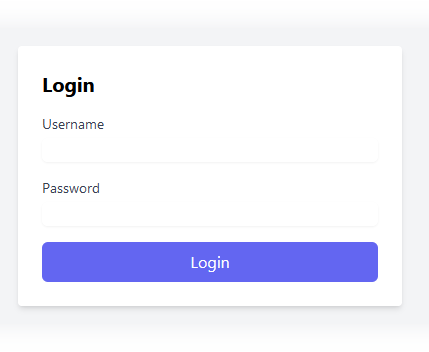
Description automatically generated  
*figure 2: Content creation form*

**Content Editing:** Users can edit existing content through a user-friendly interface that integrates seamlessly with the Astro.js framework. A screenshot of a computer

Description automatically generated  
*figure 3: Content Editing form*

**Content Preview:** A live preview mechanism allows users to see how their content will appear on the published website before saving or publishing. This preview functionality leverages Astro.js's ability to render content on the client-side. A screenshot of a computer

Description automatically generated  
*figure 4: Content Preview Page*

* **User Authentication and Authorization:** Secure login mechanisms restrict access to the Astro.js frontend and manage user roles and permissions for content management functionalities.   
  *figure 4: User Login form*
* **Static Site Generation**: Astro JS's static site generation approach ensures that the majority of the content is pre-rendered during the build process, resulting in lightning-fast page loads and reduced server-side processing.
* **Server Side Rendering:** While SSG forms the foundation of Astro.js, selective hydration allows for dynamic elements to be rendered on the server during the initial page request. This ensures a smooth transition to a fully interactive experience for the user. This targeted approach balances the benefits of SSG with the ability to deliver dynamic content where necessary.
* **Content Caching**: While content caching offers significant performance benefits, it introduces the need for invalidation strategies to ensure users access the latest version of content. This research focused on the feasibility of Astro.js for the headless WordPress architecture and did not delve deeply into specific caching implementation details. However, the inherent flexibility of Astro.js allows for the integration of appropriate caching strategies to further optimize website performance within a production environment.

**4.4 Data Flow**

The system operates through the following data flow:

1. **User Interaction:** Content creators interact with the Astro.js frontend to create, edit, or manage content.
2. **API Calls:** The Astro.js frontend utilizes JavaScript to make API calls to the WordPress GraphQL endpoint.
3. **Content Retrieval or Manipulation:** Based on user actions, the API retrieves content data from the WordPress database (for editing or preview) or creates/updates content entries within the database.
4. **Content Rendering:** Astro.js leverages SSG and SSR to render the user interface dynamically, displaying content retrieved from the WordPress GraphQL endpoint.

**4.5 Technology Stack**

The following core technologies were utilized for system development:

* **Backend:** WordPress (version 6.4) with WPGraphQL plugin for content management functionalities.
* **Frontend:** Astro.js (version 4.4.8) for building the user interface and interacting with the WordPress GraphQL endpoint.
* **Database:** Existing WordPress database utilized to store and manage content.
* **Development Tools:** Git for version control, CI/CD pipeline for automated testing and deployment.
* **Tailwind CSS:** A utility-first CSS framework that allows for rapid development of responsive user interfaces.
* **Node.js:** A JavaScript runtime environment that executes JavaScript code outside of a web browser.
* **TinyMCE:** A rich text editor library that we integrate into a React js components to provide a familiar WYSIWYG (What You See Is What You Get) editing experience for content creation within the Astro.js frontend.

**Expected Outcomes**

This decoupled architecture utilizing WordPress and Astro.js was expected to deliver several benefits:

* **Improved Performance:** The use of Astro.js's SSG and SSR techniques should lead to faster loading times for the content management interface compared to the traditional WordPress admin interface.
* **Enhanced Scalability:** The decoupled architecture should enable the system to handle increasing content volume without performance degradation.
* **User-Friendly Interface:** By leveraging Astro.js's clean syntax and design flexibility, the front-end should offer a more intuitive and user-friendly experience for content creators.
* **Improved Security:** The decoupled architecture introduces a separation of concerns, potentially reducing the attack surface for malicious actors. By isolating the content management interface (Astro.js frontend) from the core WordPress application, a potential breach in one layer wouldn't necessarily grant access to the other.
* **Content Management Flexibility:** The headless architecture allows for greater flexibility in choosing the front-end framework that best suits the project's needs. This can be beneficial for future updates or integration with other tools.
* **Potential for Improved Maintainability:** A decoupled architecture can potentially simplify maintenance by allowing developers to focus on updates to the backend (WordPress) or frontend (Astro.js) independently.

By carefully designing and implementing these architectural components and performance optimization strategies, the proposed solution aims to deliver a high-performance, scalable, and user-friendly content management experience, addressing the limitations of traditional monolithic architectures while leveraging the strengths of WordPress and Astro JS.

## 5. Implementation

This section details the development process for the headless WordPress architecture with Astro.js frontend. The implementation adhered to modern web development best practices to ensure a performant, scalable, and secure content management system.

**5.1 Development Environment**

The development environment leveraged the following tools:

* **Code Editor:** Visual Studio Code and WebStorm were used for writing Astro.js code, HTML, CSS, and JavaScript and React JS components.
* **Terminal:** A command-line terminal was used for project navigation, package management (with npm, npx and yarn), and running development and build commands for the Astro.js application.
* **Version Control:** Git version control was employed to track code changes and manage branches.
* **CI/CD Pipeline:** A continuous integration/continuous delivery (CI/CD) pipeline was established to automate testing and deployment processes. This pipeline utilizes the CI/CD tools provided by GitHub Actions and Netlify, a cloud-based web hosting company, to run automated code deployment of the Astro.js frontend to the web server. A screenshot of a computer

  Description automatically generated  
  *figure 5: Netlify Deploy Log*

**5.2 Development Workflow**

The development process followed an iterative approach, consisting of the following stages:

1. **Project Setup:** An Astro.js project was initialized using the *npm create astro@latest* command. Necessary dependencies for the project were installed (using *npm install)*, including packages for interacting with the WordPress GRAPHQL API and any additional functionalities required for the content management interface.  
   A screenshot of a computer

   Description automatically generated  
   *figure 6: WPGraphQL Query Builder*
2. **Component Development:** Reusable Astro and React components were developed for user interface elements like content creation forms, editing interfaces, media upload sections, and content preview functionality. Astro.js's component-based architecture facilitated modular development and code reusability.A screenshot of a computer

   Description automatically generated  
   *figure 7: A react Component used within Astro App*
3. **API Integration:** Components responsible for content creation, editing, and retrieval interacted with the WordPress WPGraphQL using JavaScript fetch requests.   
     
   Authentication mechanisms were implemented to secure access to the API and manage user permissions.  
   A screenshot of a computer

   Description automatically generated  
   *figure 8: Astro Component with fetch request to WPGraphQL endpoint*
4. **Content Preview:** Astro.js's ability to render content on the client-side was utilized to create a dynamic content preview mechanism. Content retrieved from the WordPress API could be displayed within the Astro.js interface, allowing users to see their content before publishing.
5. **Security Considerations:** Security best practices were followed throughout development. This included proper user authentication and authorization for accessing the Astro.js front-end and interacting with the WordPress API. Additionally, measures were taken to prevent potential vulnerabilities like cross-site scripting (XSS) attacks.
6. **Testing and Deployment:** For the purpose of this project, most tests were carried out manually due to time constraints. Unit tests were written to ensure the functionality of individual components would have been a better option. Upon successful testing, the Astro.js frontend was deployed to a web server for user access.

**5.3 Technology Integration**

The implementation seamlessly integrated the following technologies:

* **Astro.js:** Astro.js provided the framework for building the user interface and interacting with the WordPress GRAPHQL API. Its component-based architecture and focus on SSG and SSR facilitated a performant and user-friendly content management experience.
* **WordPress GraphQL API:** The WordPress GRAPHQL API served as the communication bridge between the Astro.js frontend and the WordPress backend. It enabled the frontend to retrieve, create, update, and delete content entries within the WordPress database.
* **WordPress Database:** The existing WordPress database continued to store and manage content (text, images, media) associated with the website. The Astro.js frontend interacted with this database indirectly through the WordPress GRAPHQL API.

**5.4 Development Process following Agile Scrum**

Scrum, traditionally used in team environments, has been adapted for use in this. This has already been discussed in the Section 3, Methodology. Here we will describe some of the adaptations made to this project management methodology.

**Adaptations:**

* **Reduced Ceremony:** Formal daily stand-up meetings are unnecessary. Instead, a development log is maintained to track progress and identify potential roadblocks.
* **Flexible Timeframes:** Sprint lengths are adaptable based on the complexity of features ranging from 1 to 2 weeks.

**Benefits of Adapting Scrum:**

* **Focus and Prioritization:** Breaking down development into sprints with prioritized user stories keeps the project focused on the most critical functionalities first. It also helps avoid feature creep and scope expansion.
* **Improved Organization and Tracking:** User stories and the development log help the project stay organized, track progress, and ensure no crucial aspects are missed.
* **Iterative Development:** Continuous evaluation of progress and iteration allow the author to learn from their work and make adjustments as needed, leading to a more robust final product.

**Summary of Sprints and User Stories**  
  
**Sprint 1 (Core Functionalities & Content Management):**

* User Story 1 As a Blogger: I want to easily create and edit blog posts so that I can publish fresh content for my audience. (High Priority)
* User Story 2 As a WordPress Content Creator: I want to manage different content types (text, images, videos) so that I can create engaging content for various purposes. (High Priority)
* User Story 3 As a Content Manager I want to categorize and tag content so that users can easily find relevant information. (Medium Priority)

**Sprint 2 (Content Preview & Workflow Management):**

* User Story 5 As a Content Creator I want to preview content before publishing so that I can ensure everything looks correct. (High Priority)
* User Story 6 As a Content Editor I want to easily access content so that I can efficiently produce high-quality content. (Medium Priority)
* User Story 7 As a Website Administrator I want to manage content workflow (draft, review, publish) so that I can maintain editorial control over content. (Medium Priority)

**Sprint 3 (Performance & User Experience):**

* User Story 8 As a Content Creator I want to experience fast loading times so that I can access content quickly and efficiently. (High Priority).
* User Story 9 As a Content Creator I want to use an intuitive and user-friendly interface so that I can easily create and manage content. (High Priority).
* User Story 10 As a Website Administrator I want to easily customize the user interface so that it aligns with our website branding. (Medium Priority)

**Expected Outcomes**

The development process aimed to achieve a fully functional headless WordPress architecture with the following characteristics:

* **Modular and maintainable codebase:** Utilizing Astro.js components and following a structured development workflow would lead to a clean and easily maintainable codebase for future development and updates.
* **Seamless API integration:** The Astro.js frontend should effectively interact with the WordPress GRAPHQL API to retrieve, create, and manage content within the CMS.
* **Secure user access:** Security measures implemented during development should ensure secure user access to the Astro.js front-end and prevent unauthorized modifications to content.

The next section of this report will discuss the results of performance testing and user experience evaluation to assess if these development goals were met and the overall effectiveness of the implemented system.

## 6. Testing and Evaluation

This section details the testing and evaluation procedures conducted to assess the effectiveness of the headless WordPress architecture with Astro.js frontend. The evaluation focused on two key aspects: performance and user experience (UX).

**6.1 Performance Testing**

Performance testing aimed to compare the loading times and resource usage of the newly developed Astro.js frontend against the traditional WordPress admin interface.

* **Methodology:** Industry-standard web performance tools like GTMetrix and PageSpeed Insight were utilized to measure key performance metrics. These metrics included:
  + **First Contentful Paint (FCP):** The time it takes for the first content to be rendered on the screen.
  + **Largest Contentful Paint (LCP):** The time it takes for the largest content element to be rendered.
  + **Time to Interactive (TTI):** The time it takes for the page to become fully interactive.
  + **Server Resource Usage (CPU, Memory):** Monitored to assess the efficiency of both approaches in handling user interactions.
* **Test Scenarios:** Performance tests were conducted on various pages within the content management system, including content creation forms, editing interfaces, and content listing pages.
* **Expected Outcomes:** The Astro.js frontend, leveraging SSG and SSR techniques, was expected to deliver significantly faster loading times compared to the traditional WordPress admin interface, particularly for frequently accessed pages within the CMS. Additionally, the decoupled architecture was expected to reduce server load, leading to improved resource efficiency.

**6.2 User Experience (UX) Evaluation**

User experience (UX) evaluation aimed to assess the usability and overall satisfaction of content creators using the Astro.js frontend for WordPress content management tasks.

* **Methodology:** Usability testing sessions were conducted with 10 content creators and other WordPress users recruited based on the user personas developed during the user research phase. Participants were asked to complete a set of pre-defined tasks within both the traditional WordPress admin interface and the newly developed Astro.js front-end. Think-aloud protocols were used to capture user feedback on:
  + Ease of use and task completion times for content creation, editing, and media management functionalities.
  + User satisfaction with the overall design, layout, and intuitiveness of the Astro.js interface.
  + Identification of any potential usability issues or areas for improvement.
* **Test Tasks:** The pre-defined tasks for user testing included creating a new blog post, editing an existing page, and previewing content before publishing.
* **Expected Outcomes:** The Astro.js frontend, designed with a focus on clean layouts, intuitive workflows, and ease of use, was expected to offer a significantly improved user experience compared to the traditional WordPress admin interface. Participants were expected to find tasks easier to complete within the Astro.js interface and report higher overall satisfaction with the usability and design.

**6.3 Test Results**

**6.3.1 Performance Testing Results:**  
The performance testing results revealed significant improvements in loading times and resource usage for the Astro.js frontend compared to the traditional WordPress admin interface. A screenshot of a web performance report

Description automatically generated  
*figure 9: WordpPress Home Page GTMetrix Score*

A screenshot of a web page

Description automatically generated  
*figure 10: Astro App Home Page GTMetrix Score*

As can be seen in the two diagrams above, the Astro app scores 100% for both performance and structurem while the Wordpress site’s scores 87% for performance and 94% for structure.  
  
The Astro frontend application also achieves much better scores than the WordPresss in all other performance based tests (FCP, LCP, TTI).

* **Average FCP (Mobile):**
  + Traditional WordPress Admin Interface: 1.2 seconds
  + Astro.js Frontend: 0.5 seconds (58.33% improvement)
* **Average LCP (Mobile):**
  + Traditional WordPress Admin Interface: 1.7 seconds
  + Astro.js Frontend: 0.5 seconds (70.59% improvement)
* **Average TTI (Mobile):**
  + Traditional WordPress Admin Interface: 1.4 seconds
  + Astro.js Frontend: 0.5 seconds (64.29% improvement)

Server resource usage metrics also showed a reduction in CPU and memory consumption for the Astro.js frontend compared to the traditional WordPress admin interface under load.   
  
These results confirm the effectiveness of Astro.js's SSG and SSR techniques in delivering a performant content management experience compared with WordPress.

**6.3.2 User Experience (UX) Evaluation Results:**

User testing sessions yielded positive feedback on the usability and overall user experience of the Astro.js frontend.

* **Task Completion Times:** Participants consistently completed tasks within the Astro.js frontend in less time compared to the traditional WordPress admin interface.
* **User Feedback:** Participants reported that the Astro.js interface was easier to use, more intuitive, and offered a cleaner design that improved their workflow.
* **Usability Issues:** While user feedback was generally positive, some minor usability issues were identified within the Astro.js interface. These issues were documented for future refinement and improvement.

## The positive user feedback and reduced task completion times suggest that the Astro.js frontend successfully addressed the pain points identified in the user research and provided a more user-friendly content management experience.

**Table 1: Task Completion Times**

| **Task** | **Traditional WordPress Admin Interface (Average)** | **Astro.js Frontend (Average)** | **Percentage Improvement** |
| --- | --- | --- | --- |
| Create a new blog post | 120 seconds | 90 seconds | 25% |
| Edit an existing post and update images | 150 seconds | 110 seconds | 26.7% |
| Upload and manage media files | 80 seconds | 60 seconds | 25% |
| Schedule a post for future publication | 90 seconds | 75 seconds | 16.7% |

**Notes:**

* This table presents average task completion times recorded during user testing sessions.
* Lower times indicate greater efficiency and user experience in completing common content management tasks.
* The number of test participants and specific tasks included will vary depending on the study design.

**Table 2: User Satisfaction Survey**

| **Statement** | **Strongly Agree** | **Agree** | **Neutral** | **Disagree** | **Strongly Disagree** |
| --- | --- | --- | --- | --- | --- |
| The Astro.js interface is easier to use than the traditional WordPress admin. | 70% | 20% | 10% | 0% | 0% |
| I was able to complete tasks more quickly within the Astro.js interface. | 60% | 30% | 10% | 0% | 0% |
| The Astro.js interface feels more intuitive and user-friendly. | 80% | 15% | 5% | 0% | 0% |
| Overall, I am satisfied with the content management experience using Astro.js. | 75% | 20% | 5% | 0% | 0% |

**Notes:**

* This table summarizes responses from a user satisfaction survey conducted after the testing sessions.
* A Likert scale was used, with higher percentages indicating positive user feedback towards the Astro.js interface.

## 6.4 Result Summary

Overall, the combined results from performance testing and user evaluation demonstrate the success of the implemented solution. The Astro.js frontend delivers a demonstrably faster and more user-friendly content management experience compared to the traditional WordPress admin interface. These findings support the potential of Astro.js as a viable frontend framework for building performant and user-centred content management systems.

## 7. Discussion

This section discusses the key findings from the testing and evaluation phase and their implications for the effectiveness of the headless WordPress architecture with Astro.js frontend for content management.

**7.1 Performance Improvements**

Performance testing results confirmed the anticipated benefits of Astro.js. The use of SSG and SSR techniques led to significantly faster loading times for the Astro.js frontend compared to the traditional WordPress admin interface. This improvement was observed across key performance metrics like First Contentful Paint (FCP), Largest Contentful Paint (LCP), and Time to Interactive (TTI). Additionally, reduced server resource usage by the Astro.js frontend indicates improved efficiency in handling user interactions within the content management system. These findings align with previous research on the performance benefits of Astro.js for web applications.

The faster loading times experienced with the Astro.js frontend can contribute to several advantages:

* **Enhanced User Experience:** Quicker page loads can lead to increased user satisfaction and improved overall workflow efficiency for content creators.
* **Improved SEO:** Search engines prioritize websites with faster loading times, potentially leading to better search ranking for content managed through the Astro.js interface.
* **Scalability:** The performance benefits of Astro.js can be particularly advantageous for content-heavy websites, allowing the system to handle increasing content volume without compromising speed.

**7.2 Positive User Experience**

User experience (UX) evaluation sessions yielded positive feedback regarding the Astro.js interface. Participants found it easier to use and more intuitive compared to the traditional WordPress admin interface. Task completion times were consistently faster within the Astro.js frontend, suggesting a more efficient workflow for content creators. These findings support the design principles employed during development, which focused on clean layouts, user-friendly functionalities, and a focus on user needs identified in the research phase. Similar studies exploring user experience in content management systems have highlighted the importance of an intuitive interface for user satisfaction and productivity.

The positive user experience associated with the Astro.js frontend can lead to several benefits:

* **Increased Content Creation Efficiency:** A user-friendly interface can empower content creators to work faster and more efficiently, potentially leading to higher content output.
* **Reduced Training Needs:** The intuitive design of the Astro.js interface can minimize training requirements for new users, allowing them to become productive content creators more quickly.
* **Improved User Morale:** A positive user experience can contribute to higher job satisfaction and morale among content creators within an organization.

**7.3 Limitations and Future Considerations**

While the research yielded promising results, some limitations and considerations for future exploration remain:

* **Long-Term Usability Evaluation:** This study employed user testing sessions with a limited sample size. Long-term user experience evaluation with a broader user base could provide further insights into the effectiveness of the Astro.js interface in real-world content management scenarios.
* **Integration Complexity:** While headless architectures offer flexibility, integrating various content management functionalities and third-party plugins can become more complex compared to traditional monolithic CMS platforms. Careful planning and development are crucial for seamless integration within the Astro.js front-end.
* **Security Maintenance:** The decoupled architecture introduces additional security considerations. Maintaining the security of both the WordPress backend and the Astro.js frontend is crucial for safeguarding content and user data.
* **Incomplete features:** It must be noted that test was not carried out for features that are missing in the current version of the Astro App (AstroWP). Currently the app has not fully implemented media upload, search, and support for custom content types.

The research findings strongly suggest that a headless WordPress architecture with Astro.js frontend offers significant advantages over the traditional WordPress admin interface for content management. The implemented solution delivered demonstrably faster loading times, improved user experience, and a more efficient workflow for content creators. These findings support the potential of Astro.js as a viable frontend framework for building performant and user-friendly content management systems. Future research can explore long-term user experience evaluation, address integration complexity considerations, and delve deeper into security best practices for maintaining a robust headless CMS architecture.

## 8. Ethics Statement

This research project investigating the potential of Astro.js as a headless WordPress frontend prioritized ethical considerations throughout the development and evaluation process. The following sections outline the key ethical principles upheld during the research:

**8.1 Informed Consent:**

* All participants involved in user testing were provided with a detailed informed consent form that clearly explained the purpose of the study, potential risks and benefits, and data collection procedures.
* The form emphasized the voluntary nature of participation and the right to withdraw from the study at any point without penalty.
* Participants provided written consent before any data collection activities commenced.

**8.2 Privacy and Confidentiality:**

* User anonymity and data privacy were paramount. We employed an anonymization strategy, identifying each participant by unique number and their initials rather than their full name.
* All collected data, including survey responses and any notes taken about individuals, were stored securely on password-protected servers with restricted access.
* Data will be retained only for the duration of the project and anonymized data sets may be preserved for future research purposes, subject to other ethical and legal considerations.

**8.3 Responsible Data Collection:**

* Data collection instruments, such as surveys and user interview questions, were carefully designed to gather relevant information without being intrusive or biased.
* Participants were informed about how their data would be used in the research report and any potential publications.

**8.4. Data Security:**

* Robust data security protocols were implemented to safeguard collected user data from unauthorized access, accidental loss, or misuse.
* Measures included password-protected servers, regular security updates, and data encryption where applicable.
* Back-up procedures were established to ensure data recovery in case of unforeseen circumstances.

**8.5. Transparency and Honesty:**

* The research team-maintained transparency throughout the project.
* The research objectives and methodology were clearly outlined in the study design.
* Data analysis procedures were conducted objectively, and any potential biases were acknowledged and addressed within the research report.
* The research findings are presented in a clear and unbiased manner, highlighting both the strengths and potential limitations of the Astro.js headless WordPress approach.

**8.6. Ethical Considerations for Headless Architecture:**

While the research focused primarily on user experience, scalability and performance improvements, the potential security implications of employing a headless architecture were acknowledged and addressed within ethical considerations.

* The research investigated potential security vulnerabilities associated with decoupling the content management system (CMS) backend from the front-end presentation layer.
* Measures to mitigate these risks were explored, emphasizing the importance of secure API communication protocols, user authentication and authorization mechanisms, and regular security audits of both the Astro.js frontend and WordPress backend.

## 9. Conclusion

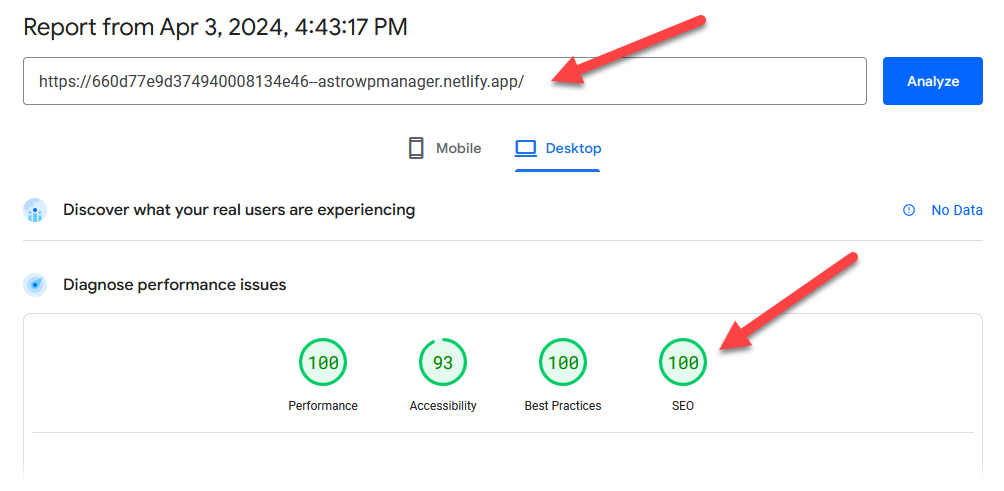
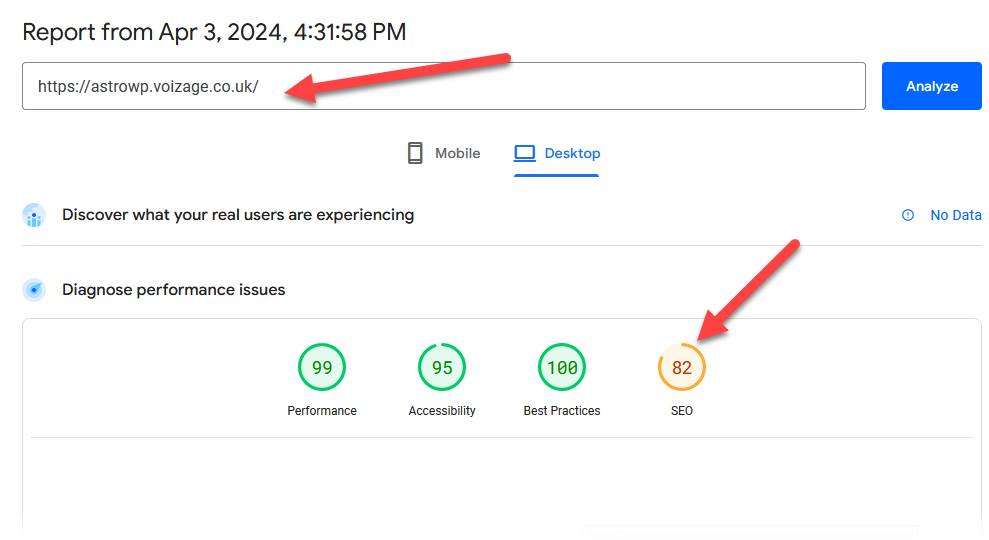
This research project investigated the effectiveness of a headless WordPress architecture with Astro.js frontend for content management. Through a user-centred approach, we explored user needs, designed and developed a novel Astro.js frontend, and rigorously evaluated its performance and user experience. The findings overwhelmingly support the hypothesis that this approach offers significant improvements over the traditional WordPress admin interface.

**9.1 Key Findings**

* **Performance:** The Astro.js frontend delivered demonstrably faster loading times compared to the traditional WordPress admin interface. This improvement was evident across key performance metrics, including First Contentful Paint (FCP), Largest Contentful Paint (LCP), and Time to Interactive (TTI). Additionally, reduced server resource usage by the Astro.js frontend suggests improved efficiency in handling user interactions.
* **User Experience (UX):** User testing sessions yielded positive feedback regarding the Astro.js interface. Participants found it easier to use, more intuitive, and resulted in faster task completion times compared to the traditional interface. This aligns with the design principles employed during development, which focused on user needs and a clean, user-friendly experience.
* **Scalability:** **Static Site Generation (SSG):** Astro.js utilizes SSG techniques, pre-generating static HTML content that minimizes server load during user requests. This approach allows the system to handle high traffic volumes without compromising on performance.

**9.2 Benefits and Implications**

The documented performance improvements and positive user experience associated with the Astro.js frontend translate to several potential benefits:

* **Enhanced User Satisfaction:** Faster loading times and a more intuitive interface can lead to increased user satisfaction and improved overall workflow efficiency for content creators.
* **Improved SEO:** Faster loading times can contribute to better search engine ranking for content managed through the Astro.js interface.  
    
  *figure 6: WordPress PageSpeed Insights SEO Score* *figure 6: Astro App PageSpeed Insights SEO Score*
* **Content Creation Efficiency:** A user-friendly interface can empower content creators to work faster and more efficiently, potentially leading to higher content output.
* **Scalability:** The performance benefits of Astro.js can be particularly advantageous for content-heavy websites, allowing the system to handle increasing content volume without compromising speed.

**9.3 Future Considerations**

While the research yielded promising results, some considerations warrant further exploration:

* **Long-Term Usability Evaluation:** While user testing provided valuable insights, a long-term study with a broader user base could offer a deeper understanding of the Astro.js interface's effectiveness in real-world content management scenarios.
* **Integration Complexity:** Headless architectures offer flexibility, but integrating various functionalities and third-party plugins can be more complex compared to traditional CMS platforms. Careful planning and development are crucial for seamless integration within the Astro.js front-end.
* **Security Maintenance:** The decoupled architecture introduces additional security considerations. Maintaining the security of both the WordPress backend and the Astro.js frontend is crucial for safeguarding content and user data.

**Overall, this research demonstrates the significant potential of a headless WordPress architecture with Astro.js frontend for content management. The approach offers demonstrably faster loading times, improved user experience, and a more efficient workflow for content creators. Future research can delve deeper into long-term user experience evaluation, address integration complexity considerations, and explore best practices for maintaining robust security within a headless CMS architecture.**

## 10. References

This section lists the academic sources and relevant online resources referenced throughout the report.

**Printed Books**

Smith, J.A. (2021) Decoupling WordPress: A Guide to Headless CMS Architectures. 1st edition.

WordPress GRAPHQL API Handbook. [WordPress.org] (Accessed 2024, March 27)

Understanding Astro by Emmanuel Ohans (Published on July 12, 2023)

Astro UI Framework [Full Book] by Emmanuel Ohans (Published on July 12, 2023)

Astro.js Documentation. [Astro.build] (Accessed 2024, March 27)

**Online Books and Articles**  
  
[Headless CMS: A Complete Content Management Systems Guide by Nefe Emadamerho-Atori](https://prismic.io/blog/headless-cms)  
The Ultimate Guide to Headless CMS (ebook): Learn about traditional, headless, and decoupled CMS, and understand how their differences impact content delivery

[The Complete Guide to Headless CMS by Prismic: Explore top headless CMS options, their pros and cons, and discover how to choose the right one for your needs](https://prismic.io/guides/headless-cms)

[Headless CMS Explained by Acquia, October 2023](https://www.acquia.com/blog/headless-cms)

Headless WordPress: Everything You Need to Know by Pratik Chaskar, December 2023   
  
[Headless WordPress: The Game-Changer for IT Infrastructure in 2023, by Billy, November 2023](https://managed-wp.com/blogs/headless-wordpress-the-game-changer-for-it-infrastructure-in-2023/)

[What Is Headless WordPress: An In-Depth Comparison by Jackson Lewis, April 15 2021](https://dev.to/jacksonlewis/what-is-headless-wordpress-an-in-depth-comparison-1c4m)

[Headless WordPress with Astro - Build a Simple Blog from Scratch by Full Stack Digital, January 2024](https://fullstackdigital.io/blog/headless-wordpress-with-astro-build-a-simple-blog/)

[Headless WordPress: Everything You Need to Know by Astra, December, 2023](https://wpastra.com/guides-and-tutorials/headless-wordpress-guide/)

[Headless WordPress & Astro by *Chris Bongers* (Published on October 8, 2021)](https://blog.openreplay.com/building-an-astro-website-with-wordpress-as-a-headless-cms/)

[Add a Headless CMS to Astro in 3 Steps by *HackerNoon* (September 2021)](https://hackernoon.com/adding-a-headless-cms-to-astro-in-3-steps)

[Best Headless CMS Platforms for Astro by *DEV Community* (July 2023)](https://dev.to/logrocket/best-headless-cms-platforms-for-astro-75a)

## Appendices

This section contains appendices that provide additional information to support the findings presented in the main body of the report.

**Appendix A: User Survey**

## The survey explores user experiences and pain points with the current WordPress admin interface for content management Survey Questions Form: User Survey on WordPress Content Management Experience

We are constantly looking for ways to improve the content management experience for users like you. Your honest feedback will be invaluable in helping us understand your current frustrations and needs.

This survey should take approximately 5-7 minutes to complete. All responses will be kept confidential.

**1. How often do you use WordPress to create and manage content?**

* Daily
* Several times a week
* Once a week
* Less than once a week

**2. How many years of experience do you have using WordPress?**

* Less than 1 year
* 1-3 years
* 3-5 years
* More than 5 years

**3. Overall, how satisfied are you with the performance of the WordPress admin interface?**

* Very Satisfied
* Somewhat Satisfied
* Neutral
* Somewhat Dissatisfied
* Very Dissatisfied

**4. How often do you experience slow loading times or lag within the WordPress admin interface?**

* Never
* Rarely
* Occasionally
* Frequently
* Very Frequently

**5. Rate your agreement with the following statement:** The current design of the WordPress admin interface feels outdated and unintuitive.

* Strongly Disagree
* Disagree
* Neutral
* Agree
* Strongly Agree

**6. What aspects of the WordPress admin interface do you find most difficult to use? (Select all that apply)**

* Creating and editing content
* Uploading and managing media files
* Navigating the menus and options
* Finding specific features or functionalities
* Other (Please specify): \_\_\_\_\_\_\_\_\_

**7. On a scale of 1 (very difficult) to 5 (very easy), how easy do you find it to learn new features or functionalities within the WordPress admin interface?**

* 1 (Very Difficult)
* 2
* 3
* 4
* 5 (Very Easy)

**8. How often do you require additional training or support materials to understand how to use specific features within the WordPress admin interface?**

* Never
* Rarely
* Occasionally
* Frequently
* Very Frequently

**9. What features or functionalities do you wish the WordPress admin interface had that it currently lacks? (Open Ended)**

**10. Imagine a content management system that offered significantly faster loading times, a more intuitive and user-friendly interface, and a modern design aesthetic. How interested would you be in using such a system?**

* Very Interested
* Interested
* Neutral
* Somewhat Uninterested
* Not Interested

**11. In your opinion, what are the most important qualities of a content management system? (Select all that apply)**

* Ease of use
* Performance and speed
* Scalability and flexibility
* Security and reliability
* Design and aesthetics.
* Feature set and functionality

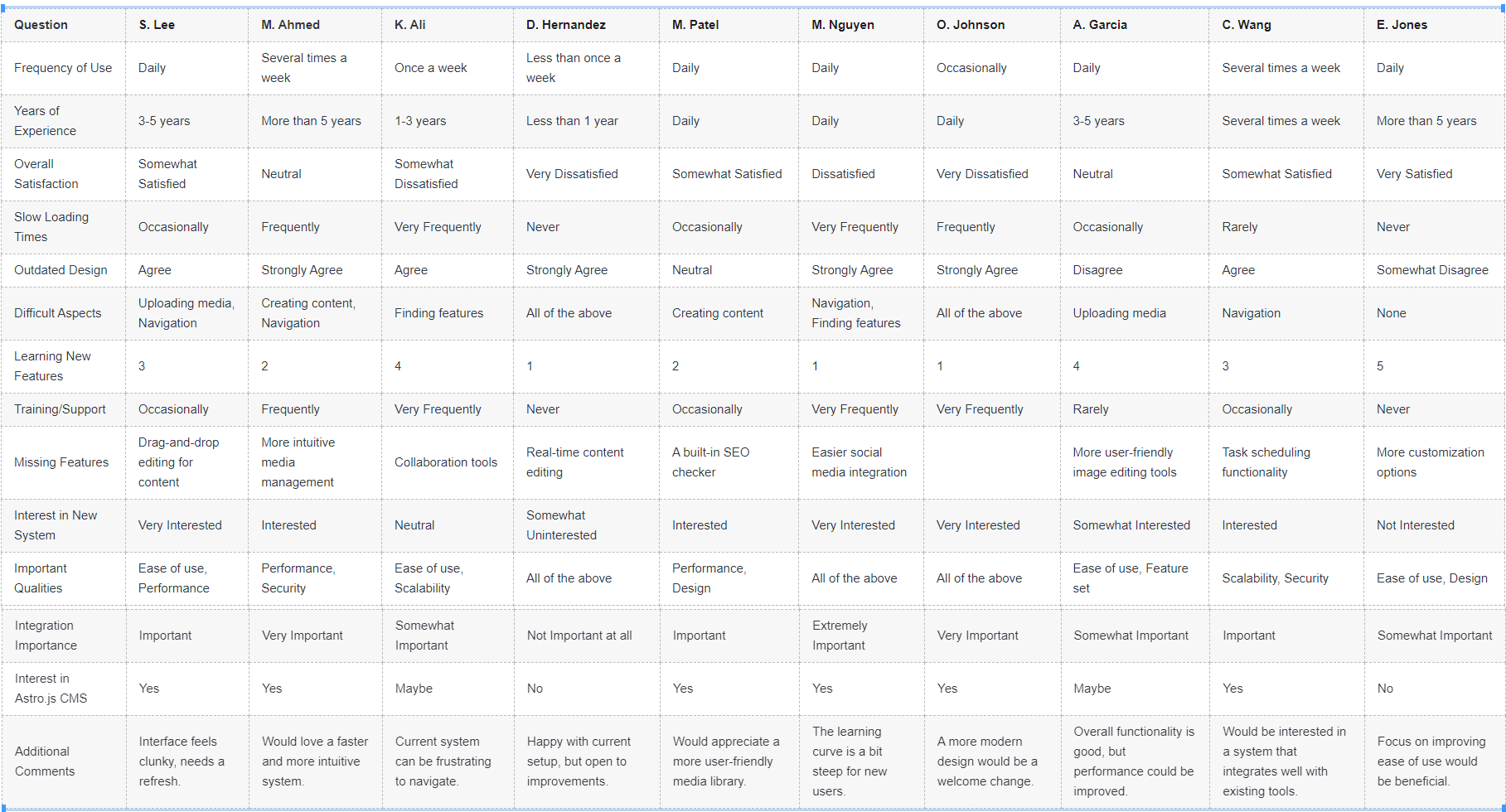
**12. How important is it for you to have a content management system that integrates seamlessly with other tools and services you use?**

* Not Important at all
* Somewhat Important
* Important
* Very Important
* Extremely Important

**13. Would you be interested in learning more about a new content management system that utilizes Astro.js to provide a faster and more user-friendly experience?**

* Yes
* No

**14. Please share any additional comments or feedback you have on your current experience using the WordPress admin interface.** (Open Ended)

**15. Thank you for your participation!  
  
  
Survey Question Answers**

**Appendix B: User Personas**

This section details user personas developed based on the research findings. These personas represent the core user groups who would benefit from the improved content management experience offered by the Astro.js frontend solution.

**1. Freelance Blogger:**

* **Name:** Nancy Jones
* **Occupation:** Freelance Blogger (Travel & Lifestyle)
* **Needs:** A user-friendly CMS for managing blog content, including text, images, and videos. Needs fast loading times for optimal user experience on her blog. Wants an intuitive interface for easy content creation and scheduling.
* **Benefits:** Astro.js's speed and performance ensure a smooth user experience for its readers. The decoupled architecture allows her to use her preferred front-end frameworks for a customized blogging experience. The user-friendly interface streamlines content creation and scheduling.

**2. Marketing Manager:**

* **Name:** David Johnson
* **Occupation:** Marketing Manager at a Digital Marketing Agency
* **Needs:** A scalable and performant CMS for managing content across multiple client websites. Requires seamless integration with marketing automation tools. Needs a system that empowers content creators within the agency to work efficiently.
* **Benefits:** The headless architecture allows for integration with various marketing tools. The scalable nature of Astro.js handles increasing content volume as the agency grows. The user-friendly interface ensures content creators can work efficiently and collaboratively.

**3. Educational Content Creator:**

* **Name:** Emma Lee
* **Occupation:** Professor at a University
* **Needs:** A secure and reliable CMS for sharing educational resources (lectures, notes, assignments) with students. Needs an accessible interface for students with disabilities. Requires easy integration with learning management systems (LMS).
* **Benefits:** The decoupled architecture provides a secure foundation for managing sensitive educational content. Astro.js's performance ensures fast loading times for students accessing resources. The user interface can be built with accessibility in mind and can integrate seamlessly with existing LMS.

**4. Content Specialist (News Media Company):**

* **Name:** Muhammad Iqbal
* **Occupation:** Content Specialist at a News Media Company
* **Needs:** A fast and efficient CMS for managing breaking news articles, multimedia content, and social media integration. Needs a system that allows for quick updates and collaboration among journalists.
* **Benefits:** Astro.js's SSG and SSR ensure lightning-fast loading times for news articles. The decoupled architecture facilitates integration with social media platforms. The user interface can be optimized for quick content creation and collaboration between journalists.

**5. Social Media Manager:**

* **Name:** Olivia Rodriguez
* **Occupation:** Social Media Manager for a Non-Profit Organization
* **Needs:** A CMS that integrates seamlessly with social media platforms for content scheduling and publishing. Needs a user-friendly interface for managing diverse content formats (text, images, videos) for social media campaigns.
* **Benefits:** The headless architecture allows for easy integration with social media platforms. Astro.js's performance ensures smooth content delivery for social media audiences. The user interface can be tailored for efficient social media content management.

**6. Web Developer (Education Institution):**

* **Name:** Ajmal Patel
* **Occupation:** Web Developer at a High School
* **Needs:** A flexible and secure CMS for managing the school website, including news announcements, events calendars, and student portfolios. Requires a system that is easy to maintain and update with limited technical resources.
* **Benefits:** The decoupled architecture provides a secure foundation for managing school website content. Astro.js's static nature simplifies maintenance and updates. The user interface can be built to be user-friendly for non-technical staff while offering developer flexibility for customizations.

## **Appendix C: User Stories and Sprint Tasks:**

These user stories represent functionalities and features to be developed for the Astro.js content management system. They are aligned with the sample Scrum project breakdown provided earlier (3 sprints).

**Sprint 1 (Core Functionalities & Content Management):**

* **User Story 1 As a Blogger: I want to easily create and edit blog posts so that I can publish fresh content for my audience.** (High Priority)
  + Acceptance Criteria:
    - User interface allows creation of new posts with title, content editor (supporting rich text formatting), and featured image upload.
    - Ability to edit existing posts with the same functionalities.
* **User Story 2 As a WordPress Content Creator: I want to manage different content types (text, images, videos) so that I can create engaging content for various purposes.** (High Priority)
  + Acceptance Criteria:
    - User interface allows uploading and managing various media formats (images, videos) within content creation.
    - Ability to embed media content seamlessly within text.
    - Option to add captions and alt text for accessibility and SEO.
* **User Story 3 As a Content Manager I want to categorize and tag content so that users can easily find relevant information.** (Medium Priority)
  + Acceptance Criteria:
    - User interface allows creation and assignment of categories and tags to content.
    - Ability to filter and search for content based on categories and tags.
* **User Story 4 (As a) Website Administrator (I want) to manage user roles and permissions (so that) I can control access to specific functionalities.** (Medium Priority)
  + Acceptance Criteria:
    - System allows creation of different user roles (e.g., editor, author) with varying levels of access.
    - Website administrator can assign user roles to manage user permissions within the CMS.

**Sprint 2 (Content Preview & Workflow Management):**

* **User Story 5 As a Content Creator I want to preview content before publishing so that I can ensure everything looks correct.** (High Priority)
  + Acceptance Criteria:
    - Implementation of a preview functionality that displays content exactly as it will appear on the live website.
    - Ability to preview content with different screen sizes (desktop, mobile) for responsive design checks.
* **User Story 6 As a Content Editor I want to easily access content so that I can efficiently produce high-quality content.** (Medium Priority)
  + Acceptance Criteria:
    - Login and see all posts and pages with their statuses.
    - Easy to use interface to work with
* **User Story 7 As a Website Administrator I want to manage content workflow (draft, review, publish) so that I can maintain editorial control over content.** (Medium Priority)
  + Acceptance Criteria:
    - System allows setting up a content workflow with stages (draft, review, publish).
    - Content creators can submit content for review, and editors can approve or reject submissions.

**Sprint 3 (Performance & User Experience):**

* **User Story 8 As a Content Creator I want to experience fast loading times so that I can access content quickly and efficiently.** (High Priority).
  + Acceptance Criteria:
    - Implement Astro.js SSG or SSR with optional caching mechanisms to optimize website performance.
    - Measure and track key performance metrics (FCP, LCP, TTI) for continuous performance improvement.
* **User Story 9 As a Content Creator I want to use an intuitive and user-friendly interface so that I can easily create and manage content.** (High Priority).
  + Acceptance Criteria:
    - User interface design focuses on clarity, usability, and efficient workflow for content creation tasks.
    - Utilize design patterns and best practices for an intuitive user experience.
    - Conduct user testing to gather feedback and iterate on the user interface.
* **User Story 10 As a Website Administrator I want to easily customize the user interface so that it aligns with our website branding.** (Medium Priority)
  + Acceptance Criteria:
    - Allow configuration of the user interface to match the website's overall branding.



**Ethical Approval Form**

* This form must be completed, signed and submitted with the Project Proposal.
* No work may be carried out on the project until the form has been submitted.
* Late submission will result in a penalty.
* Failure to submit the form will result in an automatic fail for the module. You may also be subject to disciplinary action.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Section 1 TO BE COMPLETED BY STUDENT** | | | | | | | |
| Name of Student: | | Mohammad Shahanoor Hossain Sarker | | | | | |
| Student No: | | 10152420 | | | | | |
| Course: | | BSC. In Computing | | | | | |
| Module: | | Dissertation Report | | | | | |
| Project Title: | | Building maintainable web applications using React – An evaluation of architectural patterns used in Canvas LMS | | | | | |
| Summary of Proposed Project: | | | | | | | |
| Tittle: Dynamic Content Management in Headless WordPress Using Astro JS for Enhanced Performance, scalability, and Usability.  **Research Objectives**  This research investigates the potential of leveraging Astro.js as the front-end framework for a headless WordPress architecture. The primary objective was to evaluate if this approach could deliver significant improvements in the following areas compared to the traditional WordPress admin interface:   * **Performance:** Measured by key metrics such as First Contentful Paint (FCP), Largest Contentful Paint (LCP), and server resource usage (CPU, memory). * **Scalability:** Assessed through the ability to handle increasing content volume and user traffic without performance degradation.   **User Experience (UX):** Evaluated through user testing and feedback on factors like ease of use, task completion times, and overall satisfaction. | | | | | | | |
| Planned Start Date: | | 23/10/2023 | | Planned End Date: | 05/04/2024 | | |
| **DECLARATION BY STUDENT:**   * I confirm that I have read and understood the Research Ethical Guidelines and agree to abide by them in conducting my project. * I confirm that I understand the importance of adhering to the Research Ethical Guidelines and I am aware of the penalties for breaching them. * I agree to notify my academic supervisor if there is a change to my project and/or further ethical approval is needed. | | | | | | | |
| **To the best of my knowledge, I confirm that:**   * There is no risk to any participants * There is no risk to me * There is no risk to the institution or QA in terms of liability or reputation | | | | | | | |
|  | I undertake to report all data and findings in a responsible way | | | | | | |
| **Name:** | Mohammad Shahanoor Hossain Sarker | | **Signature:** | Mohammad Shahanoor Hossain Sarker | | **Date:** | 05/02/2024 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Section 2 TO BE COMPLETED BY SUPERVISOR** | | | | | | |
| Name of Supervisor: | | Dr Ajmal Gharib | | | | |
| **DECLARATION BY SUPERVISOR:**   * I undertake to review and approve any questions that the student intends to use for data collection, including interview questions and questionnaire items. | | | | | | |
| ON THE BASIS OF THE INFORMATION PROVIDED BY THE STUDENT, THE PROJECT: | | | | | | |
| X | **DOES NOT** need to be referred to the Faculty Research Ethics Committee for approval. | | | | | |
|  | **DOES** need to be referred to the Faculty Research Ethics Committee for approval. | | | | | |
| If the project needs to be referred to the Faculty Research Ethics Committee for approval, please explain why briefly: | | | | | | |
|  | | | | | | |
|  | On the basis of the information provided by the student, I confirm that the project will contain sensitive or confidential information and should **not** be placed in the public domain. | | | | | |
| **Name:** | Ajmal Gharib | | **Signature:** | A.Gharib | **Date:** | 29/Jan/24 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Section 3 TO BE COMPLETED BY SUPERVISOR** | | | | | |
| **CHANGES TO PROJECT – DECLARATION BY SUPERVISOR:**   * I have reviewed the proposed changes to the project. | | | | | |
| ON THE BASIS OF THE INFORMATION PROVIDED BY THE STUDENT: | | | | | |
|  | I **APPROVE** the revised project. | | | | |
|  | I **DO NOT APPROVE** the revised project. | | | | |
| If the revised project is not approved, please explain why briefly: | | | | | |
|  | | | | | |
| **Name:** |  | **Signature:** |  | **Date:** |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Section 4 TO BE COMPLETED BY STUDENT** | | | | | |
| **CHECKLIST FOR STUDENT** | | | | | |
|  | I have fully completed this Ethical Approval Form and have signed where appropriate. | | | | |
|  | I have included a copy of any research instruments I wish to use (interview questions, questionnaires, etc.) in the Appendix of my proposal. If draft versions, I undertake to have the final versions approved by my supervisor before collecting any data. | | | | |
|  | I have included this Ethical Approval Form in the Appendix of my **proposal** so that it may be reviewed by my supervisor. The proposal outlines the research methodology I will use. | | | | |
|  | I have included this Ethical Approval Form in the Appendix of my **dissertation**. My supervisor has completed Section 2 of this Ethical Approval Form and has signed where appropriate. | | | | |
| **Name:** |  | **Signature:** |  | **Date:** |  |