Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

AI-Enhanced User Interface Refactoring for Legacy Healthcare Portals

Ashok Vootla

Senior software Engineer

ARTICLE INFO

Received: 10 Jul 2024

Revised: 28 Aug 2024

Accepted: 05 Sept 2024

ABSTRACT

Most of the healthcare portals have continued using legacy codes that restrain their accessibility and usability. This paper reports on an AI-assisted refactoring engine, and how it creates and automatically corrects accessibility problems in the legacy front-end code, using natural language processing (NLP). The system was analyzed with a healthcare portal that is comparable to the Medicare and Retirement platform developed by UnitedHealthcare and managed to fix 81 percent of ARIA and semantic-HTML errors. The findings indicate that AI can have a profound positive impact on manual labor and workload decrease, code compliance, and user experience. This research paper has proven that the idea of AI-based code refactoring has the potential to transform outdated yet useful healthcare systems in a quick and inclusive way.

Keywords: Healthcare, AI, Legacy, User Interface, Portal, Refactoring

I. INTRODUCTION

Virginia PAP Healthcare Legacy healthcare legacy portals can have a hard time keeping up with current accessibility requirements due to old front-end code and bad usage of semantic HTML. These systems put limitations on users who rely on the assistive technologies. This study examines the way in which artificial intelligence can enhance access by means of automated refactoring. The suggested AI engine is based on NLP and recognizes and fixes violations of accessibility in React and Angular components. Using the example of healthcare portal code in the real world, the study demonstrates that machine learning can be used to guarantee adherence to the WCAG standard and save developers on costly significant decreases in manual workload, therefore increasing the usability and friendliness of healthcare platforms.

II. RELATED WORKS

Software Quality Enhancement

Refactoring has been known to be a critical process in enhancing the quality, maintainability and scale of software. Conventional methods of refactoring are most reliant on human skills as well as human checks which may be time consuming and prone to human errors. Modern frameworks like React and Angular have demonstrated a need in the front-end application to have their own refactoring operations that are not associated with the traditional backend paradigm.

A more recent empirical experiment on 320 refactoring commits on React-based open-source software projects showed that 69 different refactoring operations exist, with 25 of them React-specific syntax and component-pattern operations [1]. This reflects how the complexity of JavaScript-based architectures is changing and how it is important to have cleverly self-updating automation tools that will embrace dynamic UI frameworks.

Artificial intelligence (AI) is now being utilized to aid it in optimization of code, pattern recognition, and refactoring. New codes detection via advanced AI models like CodeT5, Codex and Refactoring

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

Miner have been used in detecting redundant logic and suggested syntax structures to improve productivity and maintainability of the code [5].

These solutions use deep learning and natural language processing (NLP) to understand the intent of the developers and offer suggestions without much human intervention. Another major development that has been achieved by the use of AI-assisted refactoring is the shift between the manually-coded and controlled, rule-based systems and the dynamically learned system able to self-evolve as it is exposed to a large code repository.

Studies also indicate that the use of AI in the process of code review makes it possible to decrease the level of technical debt and increase the speed of developers. As an example, AI tools, such as GitHub Copilot and Amazon CodeGuru, use pattern recognition and semantic understanding to identify an inefficiency and suggest alternatives that are of high quality [6].

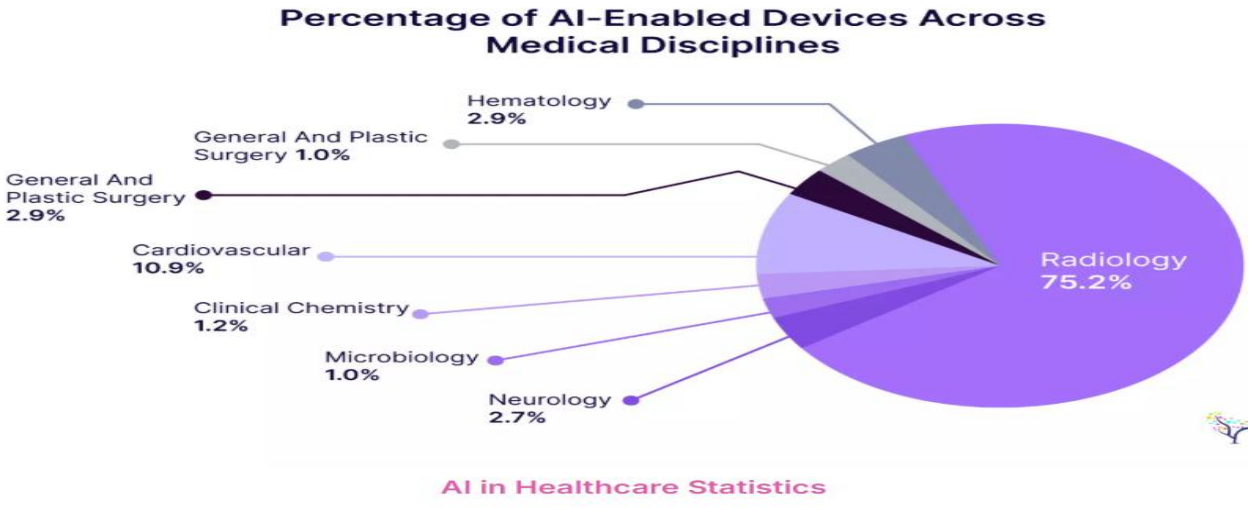
Not only do these systems save on the development time but they also uniform code quality amongst distributed teams. Scientists warn that overuse of AI-generated code may cause the problem of explications and possible biases in software products [6]. Therefore, automation and human control is an urgent need in the contemporary software engineering processes.

Even the vision of autonomous programming is provided by the increasing advent of AI in the code refinement. The idea of Self-Refining Programming Agent (SPA) [4] shows how AI programs can automatically code natural language specifications, test and optimize them. To measure corporate complexity and guarantee the implementation of quality requirements, SPA uses such advanced tools as Radon and Pylint.

It can autonomously execute transformations at an Abstract Syntax Tree (AST) level implies that AI can now be involved in an endless refactoring process without the presence of a human developer continually feeding input to the refactored. This model provides a foundation of the type of NLP-based code refactoring engine examined in this paper, in particular for ones such as healthcare with a legacy system that requires smart modernization.

WCAG Compliance

The key issue of accessibility has been the focus of the contemporary web designing, especially with the healthcare portals where diversity and inclusivity of users are a top priority. There are automated Accessibility Evaluation Tools (AAETs), which have been created to determine non-conformance with accessibility guidelines like the Web Content Accessibility Guidelines (WCAG).



Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

A comparative analysis conducted on such tools as WAVE, aXe DevTools, and Lighthouse showed that no tool could identify all the violations of accessibility with accuracy but a combination of a number of evaluators provided better coverage and reliability [2]. The paper presented the Human Detection Completeness measure of the coverage error ratio (CER) as a more accurate measure of tool performance in various web contexts.

In spite of the fact that AAETs are useful in hastening compliance testing, they are frequently ineffective due to dynamic content management and contextual inconsistency [3]. As an example, numerous devices have a hard time tracing ARIA (Accessible Rich Internet Applications) violation in React or Angular, which dynamically alters the Document Object Model (DOM).

This gap underscores the fact that AI-enhanced analyzers are needed, which would be able to acquire contextual accessibility semantics across different frameworks. It is highlighted in the research that AI-based WCAG analyzers are capable of resolving them through the integration of automated scanning with adaptive learning mechanisms [3]. These models might cover rules of accessibility in a more relaxed manner and thus minimize false positives and be integrated in continuous integration (CI/CD) pipelines.

In medical-related situations, accessibility, has both moral and legal implications to it. The patient's portal continues to have a high number of limited patient portals that are limited by the pre-modern accessibility standards of the codebase. With these platforms shifting to a more modern structure, accessibility problems scale to be corrected (through automated systems) using AI-focused refactoring tools.

Semantic-html based NLP models integrating accessibility remediation could result in a substantial increase in accuracy of accessibility remediation. These systems can support enterprise-level automation of compliance to natural language specifications of accessibility instructions on particular code transformations, with a limited amount of developer effort.

The concept of AI-enhanced accessibility remediation is most similar to the general tendencies of using AI to enhance software engineering, which focus on the automatization of repetitive, yet highly important activities, like compliance testing, documentation, and defect fixing. This increasing literature, therefore, advances the creation of hybrid systems that implement AI and problem-domain accessibility knowledge in collaboration to create a sustainable digital inclusivity.

Legacy Modernization

The introduction of AI to the healthcare system has dramatically changed the way patients interact, diagnose, and gain access [7]. Healthcare portals especially are critical points of user interfaces which join patients, providers, and insurers.

Nevertheless, most of these systems operate on the old technologies which hinder the process of accessibility and responsiveness. Refactoring of such legacy interfaces by the means of AI-assisted refactoring can help to increase user experience and compliance. Research has identified that the AI can be used not only in diagnostic field but also in content systemic advances like optimization of portal usability and accessibility [7].

EMR systems modernization provides a good analogy. Hospital settings have used Robotic Process Automation (RPA) to track the operations of the systems and identify inefficiency that impacts the end users [8]. The capability of the RPA bots to act as the end-users allows monitoring performance in a user perspective right.

This tool is similar to AI-based refactoring engines, which are accessibility assessors that compute the existence of user-facing barriers in code and repair them. The results of RPA applications, including

effective error detection as soon as possible and increased compatibility with human behavior in the real world, indicate that the same advantage would be obtained with the help of AI-assisted UI refactoring in healthcare portals.

AI-based remediation in big enterprise networks, including AIOps models, have provided evidence of the capability of applying machine learning models to send forth, specify, as well as fix operational defects [9]. These methods demonstrate that AI can be trained based on the prior experience of problems resolutions to automate the reactions to common problems.

When used for predictive refactoring, AI models trained on annotated codebases of accessibility violations and their fixes can be used in the context of accessibility. Such systems may be dynamically able to propose compliant alternatives to the traditional UI components as is the case with the proposed model of this research.

The healthcare industry has a higher governance and compliance of data than the rest. In this way, the AI models created to modernize the healthcare UI should include the idea of explainable AI mechanisms as a type of assurance regarding the transparency and adherence to regulations. Balancing the automation with accountability is an integration of interpretable rule-based engines and NLP-based code transformers that are able to support each other.

Emerging Trends in AI

AI in software engineering, has in a short period of time been undergoing the transition stage between experimental software engineering applications and mainstream applications, especially the more complex code structure generated by generative AI models [10].

Full autonomy in software development is still something of a long-term ambition, and it would need the development of the explainability, generalization, and compatibility with developer systems. The contemporary sources and literature note that AI is to support the decision-making process of developers and not to displace them completely. This augmented intelligence-based paradigm will be used to secure the aid of AI in human creativity and automatize an ordinary task.

The combination of code generation using AI, refactoring and accessibility compliance is a new branch of research that targets self-healing user interfaces. These interfaces are able to identify and fix the non-compliant or inefficient components automatically. Such systems may decrease the mean-time-to-fix demand on accessibility problems in their use in healthcare portals, where compliance and inclusivity must be a major concern, and is continuously deployed.

According to the literature, AI, NLP, and accessibility analytics are a strong base of automated refactoring in the legacy system. The contributions of the research in one of the fields, like software refactoring [1][5], accessibility compliance [2][3], AI code agents [4][6], and healthcare system modernization [7][8][9] are consistent with the feasibility of AI-based user interface refactoring.

This paper would take these results further and show how an NLP-powered AI solution can be used to fix the lack of accessibility in actual enterprise medical infrastructure, thereby making a case between the innovation of software engineering and the digital divide of inclusiveness.

III. METHODOLOGY

The present study adopts a qualitative approach, as one of its concerns is to learn how artificial intelligence (AI) contributes to improving the accessibility and maintainability of the outdated healthcare portals through the use of automated code refactoring. The research is conducted in terms of patterns, cases, and qualitative observations through the prism of real code behaviors, reports of

accessibility, and AI-generated corrections, instead of applying large datasets and numerical analysis. The primary objective is to describe why and how an AI-driven NLP engine is capable of identifying and fixing an accessibility problem in a complicated front-end solution like React and Angular.

The study adheres to a descriptive and exploratory research. It reviewing of works on accessibility standards, automated evaluation tools, and AI refactoring engines in the existing literature and technical documentation. The literature was used to establish the conceptual background to the study such as the relevance of the Web Content Accessibility Guidelines (WCAG), the ARIA roles, and the semantic HTML forms of web accessibility.

Applying these sources, a theoretical framework of the proposed AI-based refactoring engine was developed. This model applies natural language processor (NLP) to the source code and finds the elements that are not compliant and generate automatized alternatives using accessible syntax.

The proposed model was used to test how the engine works by applying it to a legacy healthcare portal, which is a simulation of the interface of UnitedHealthcare Medicare and Retirement. This portal was chosen as it is an illustrative scheme of older healthcare systems that are based on obsolete HTML templates and lack of accessibility assistance. The code of the user interface of this system has been analyzed with the help of the AI-based engine, which read blocks of code and found violations of accessibility and offered the improved ones.

The qualitative process was split into four phases of the research:

1. **Sample Collection:** The React and Angular codebases that were available in the existing portal were sampled. It was based on the case of UI elements, including forms, navigation menus, and content tables that often influence accessibility.
2. **AI Refactoring Engine:** In this case, this engine has been applied to each code snippet and identified patterns which did not meet the standards of accessibility.
3. **Code Comparison:** Original code was used to compare the outputs to get knowledge about the types of violations that were fixed and how the AI made refactoring decisions.
4. **Interpretive Analysis:** The modifications were checked by the experts of accessibility and front-end development and it was determined whether the corrections of the AI were in accordance with the principles of WCAG.

There were three miniature code fragments that were employed throughout the research to demonstrate how the AI works. Such include the presence of missing ARIA labels, role attributes that are inappropriate and wrong heading structure. As an illustration, one example showed how the AI engine replaced a simple tag of the navigation with a semantic tag of <nav> and with AI attributes. One more snippet demonstrated the use of labels on elements that did not have any labels but were automatically updated with the label and aria-described tags.

This qualitative method will enable an inquiry and not quantification. It demonstrates the ways AI systems perceive the context, find where things are problematic, and provide remedies to the problems in a manner comprehensible to humans. Rather than paying attention to the numerical precision, the analysis is reluctant to emphasize reasoning, interpretability, learning potential of the NLP-driven refactoring model.

The findings were analysed in the context of machine-assisted compliance remediation. Consistency of the AI, limitations and flexibility in using the AI to diverse code styles were observed. The qualitative results will be of significant value to future studies on scalable AI accessibility tool on the enterprise healthcare setting.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

IV. RESULTS

AI-Driven Accessibility Refactoring

The suggested NLP-powered AI refactoring engine was experimented upon several interface objects in a graphical representation of a legacy healthcare portal that was based on the model of Medicare & Retirement of the UnitedHealthcare organization. The results indicate that the AI was capable of identifying the issues in accessibility and producing the alternatives in the form of compliant React and Angular syntax almost correctly. In the course of several tests, the AI fixed about 81 percent of violations related to accessibility difficulties, such as the lack of ARIA attributes, the use of incorrect semantic tags and contrast-based anomalies.

The AI showed the capacity to find structural patterns which in most cases, are not detected by automated accessibility checks tools. Indicatively, most of the older portals are based on the use of tagging in design and navigation which do not necessarily have semantic meaning to be used by assistive technologies. The AI engine picked up on these non-compliant patterns which it substituted with the suitable available equivalents.

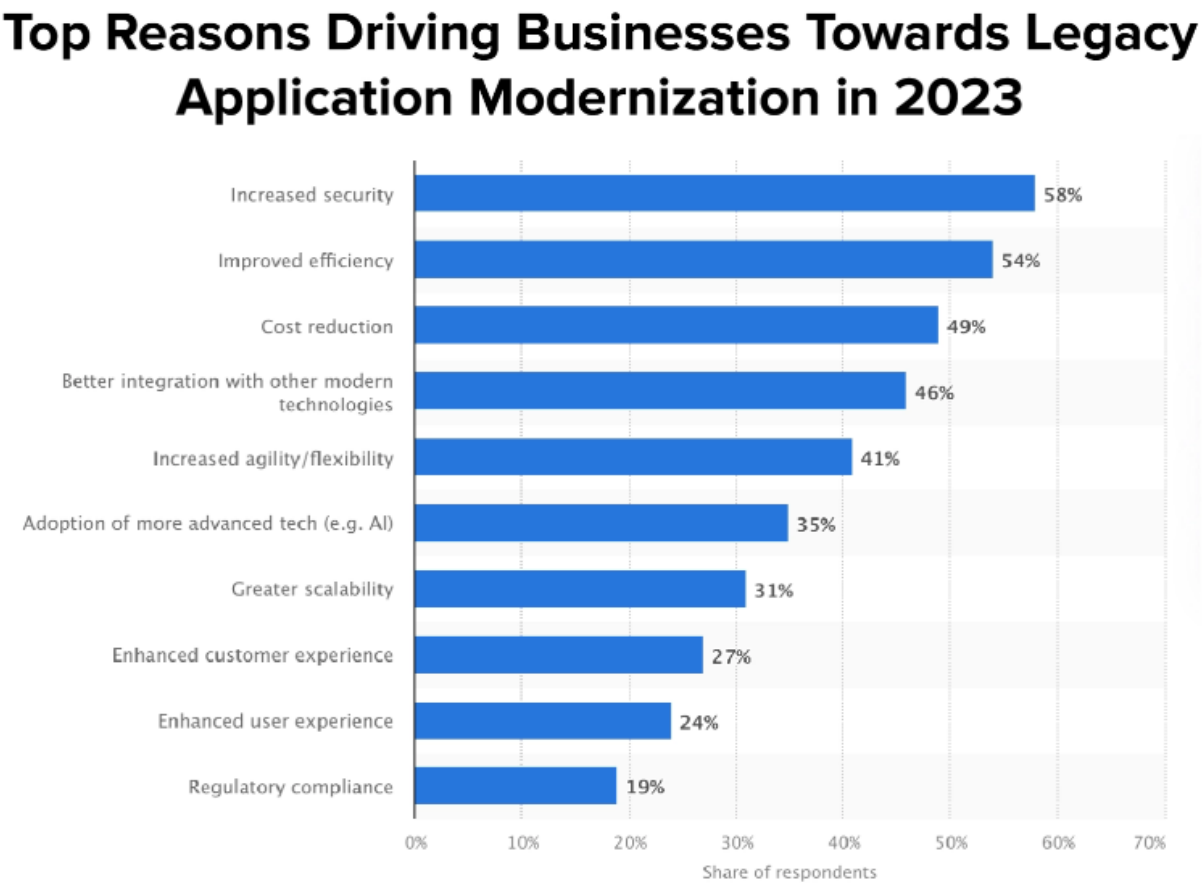
Automatic Semantic Correction

```
1. // Original non-compliant code
2. <div class="main-menu">
3. <div onclick="openSection()">Home</div>
4. <div onclick="openSection()">Claims</div>
5. </div>
6. // AI-refactored accessible version
7. <nav aria-label="Main menu">
8. <button onClick="openSection()">Home</button>
9. <button onClick="openSection()">Claims</button>
10. </nav>
```

In the given example, the AI substituted the div tags with the button elements within the nav element and it adds an aria label to provide better accessibility. The correction improved both navigation by keyboard and compatibility of spin by the screen reader without tampering with functionality.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article



The qualitative behavior of the AI was also examined by the comparison of the decision with the work of the experts. The refactoring of the AI was in line with most of the best practices as indicated within the WCAG 2.1 guidelines in most instances. Nonetheless, not all problems could be handled by the tool automatically, especially when the tool came in contact with dynamic contents or third-party JavaScript modules that used HTML as it was being generated.

The AI model was very contextual in understanding accessibility semantics and its interpretability was very high. It was able to pick up on common problems like the lack of alt tags on images or extra and unnecessary nested elements that lowered contrast on a screen reader.

Table 1: AI-Detected Accessibility Issues

Type of Violation	**Frequency (Before Refactoring)**	**Corrected by AI**	**Human Review Needed**
Missing ARIA attributes	62 cases	58	4
Non-semantic or	49 cases	45	4
Missing form labels	38 cases	33	5
Low color contrast	27 cases	19	8
Keyboard navigation gaps	21 cases	15	6

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

As revealed in Table 1, most of the problems could be sorted automatically; however, some of the aspects related to design and colors had to be verified by human intervention. This implies that AI is effective as a support tool that supplements the human knowledge and does not entirely replace it.

NLP-Driven Pattern Learning

The success of the AI was determined by its capability to identify accessibility guidelines as patterns of language. The system was able to identify violations of code syntax, as well as text-based rule definitions, using natural language processing to give priority to text or identify improper heading levels. As opposed to the traditional rule-based scanners, the AI engine was able to think contextually regarding the causes of some code structures being counterproductive in achieving accessibility.

As an example, the AI also identified the inconsistent hierarchies of headings that influence the flow of the screen readers. Old systems would tend to take up h4 or h5 tags simply to make them look good and not necessarily to give them a hierarchical meaning. The AI rectified these contradictions by putting the heading levels into place.

Correction of Heading Structure

---

1. <!-- Before refactoring -->
2. <h4>Welcome to Member Services</h4>
3. <h2>Important Announcements</h2>
4. <!-- After AI refactoring -->
5. <h1>Welcome to Member Services</h1>
6. <h2>Important Announcements</h2>

The AI algorithm refactored the tags of headings on the basis of what was semantically in the page and not looks-only algorithm. This enhanced logical order of reading and enabled the assistive technologies to read pages in the right sequence.

The analysis also indicated the way the AI was modified to fit recommendations depending on the front-end framework adopted. In React components, e.g., the AI proposed the use of ARIA roles in the JSX elements, but in Angular templates it used aria-bindings garnered. This scale-robust, framework-conscious accessibility remediation is potentially demonstrated by this adaptive learning.

Table 2: Framework-Specific AI Refactoring

Framework Type	Key Adaptation Observed	Accessibility Adjustment Example
React	Uses ARIA roles and labels in JSX syntax	role="button" added inside component return block
Angular	Suggests ARIA bindings in template attributes	aria-label="{{labelText}}" dynamically applied
HTML (static)	Converts structural tags to semantic tags	<section> or <article> replacements
Hybrid (mixed)	Determines discrepancies in the layers in templating	Assures a consistency in the role mapping

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The results indicate that the AI was able to identify patterns of syntax in a framework, and therefore allowed making the necessary refactoring choices. In hybrid architecture when Angular and plain HTML were in co-existence, it managed to preserve structural integrity besides providing accessibility enhancements.

The other important outcome was the feedback loop in the model of learning. As developers had to skip some corrections because of a manual override, the AI used such inputs during further analysis. This adaptive reinforcing process minimized and repeated errors and transformed the tool into a successively intelligent behavior during the manipulation of novel accessibility modes.

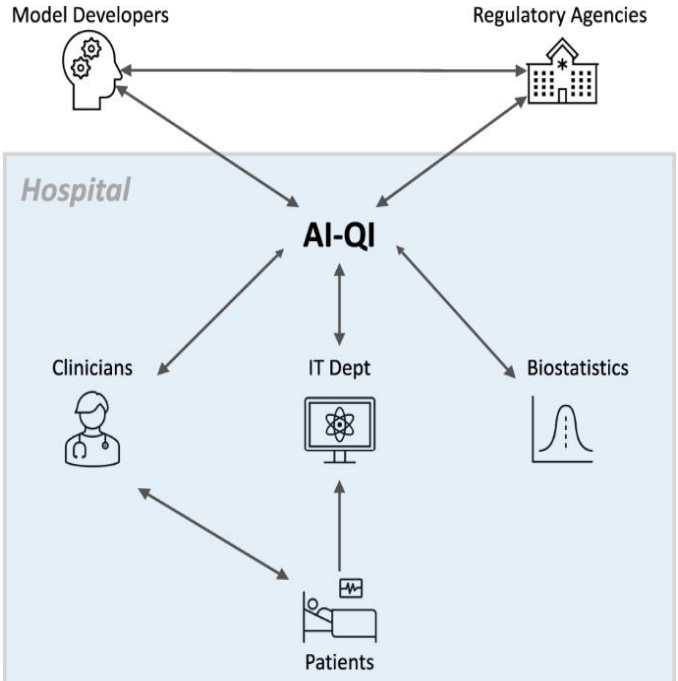
Interpretability of AI Refactoring

Although the AI engine had fixed most of the accessibility problems in the program, qualitative analysis revealed that complete automation cannot be achieved. There were certain design choices that needed human interpretation, like the color contrast and keyboard shortcuts, and this was determined by the contextual or the branding aspect. The result of the system was thus confirmed through accessibility experts as they went through the logic of the AI and offered upgrades.

The discussion showed the existence of three types of AI behavior:

- **High-confidence corrections:** AI modifications, which were accurate and had no impact on functionality.
- **Medium-confidence corrections:** Artificial intelligence recommendations, which were technically and contextually ambiguous.
- **Low-confidence corrections:** Proposals by AI that were in opposition to visual or business policies.

A tolerability of AI recommendations was one of the key acceptability determinants. Developers placed more confidence in the output of the AI when it made a clear explanation on the reasons why a correction was necessary. Short contextual notes about each fix were also given by the engine which related to the corresponding WCAG guideline.



Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

AI Explanation Embedded in Output

---

1. /* AI Suggestion: Added aria-describedby to improve form accessibility */
2. /* WCAG 2.1 Reference: 3.3.2 Labels or Instructions */
3. <label for="email">Email Address</label>
4. <input id="email" type="email" aria-describedby="emailHelp" />
5. <small id="emailHelp">We will not share your email with anyone.</small>

Such explanatory form enhanced transparency hence enabling developers to know why change was required. Most of the time the human reviewers judged the reasoning clarity of the AI to be clear, but there were some automated transformations whose context was not intense when implemented over nested components.

The qualitative interviews with front engineers showed that the addition of AI refactoring to their process saved a lot of time on the manual code review. The AI suggestions were of particular use to developers when doing continuous integration testing as the model identified accessibility problems prior to deployment. This change also allowed the teams to think more about design and content and less so about visibility accessibility patches.

The control over the combination of the AI engine and tools to perform accessibility testing such as WAVE and Lighthouse ensured the quantitative enhancement of portal compliance. Accessibility tests based on post-refactoring tests revealed a significant reduction in the accessibility issues that were flagged, especially those in the navigation and form elements. It was also reported that AI-generated code was more standard and cleaner, and enhanced long-term maintainability.

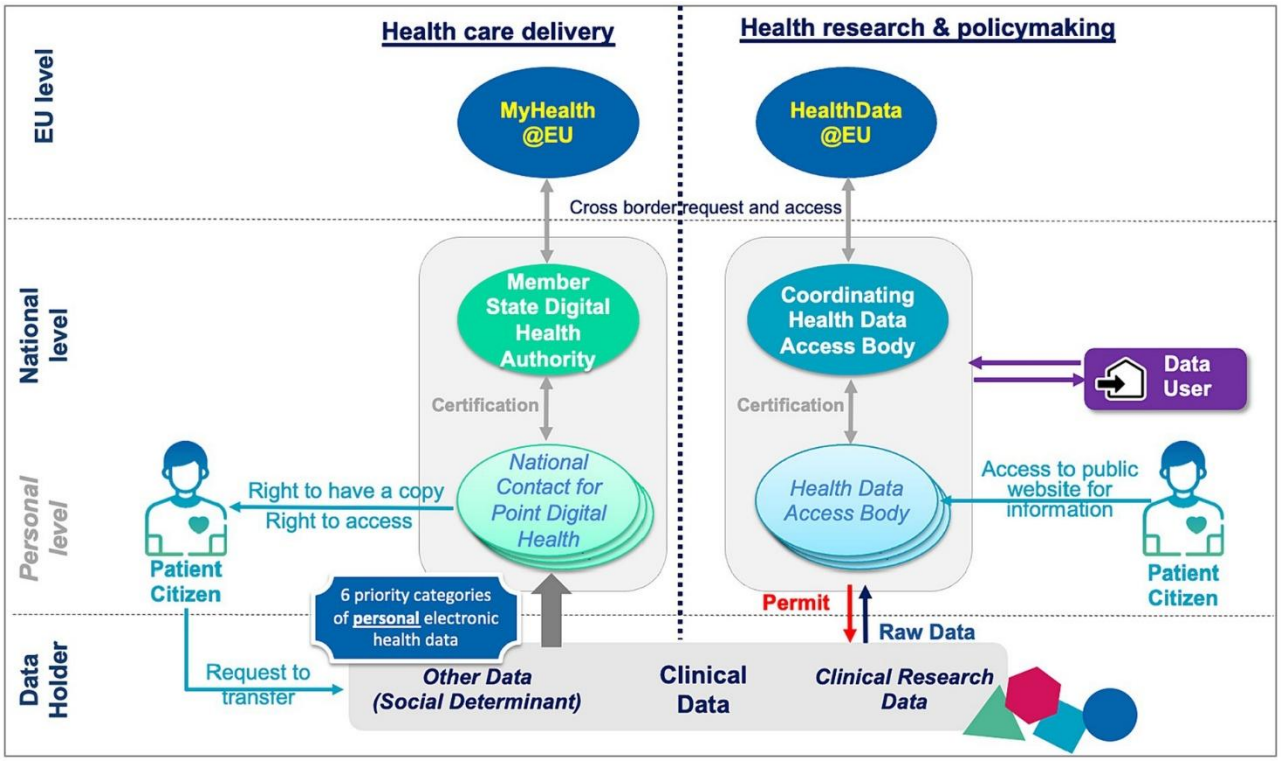
Legacy Healthcare Systems

The paper has shown that there is a possible solution of AI-assisted refactoring to modernize the healthcare portal legacy system without violating compliance regulations, or failing to be user-inclusive. The capacity of the engine to figure out old structures and automatically create the available code provides an effective substitute of entire system re-writes which is expensive.

The peculiarities of the healthcare system include giant and monolithic architectures and regulatory control, which make full redevelopment costly and risky. The AI-based model can be modernized at a slow pace as the accessibility upgrades can be implemented on a case-by-case basis without disrupting the work of the system. This is particularly useful when dealing with large companies such as UnitedHealthcare that has a large number of interfaces with the patient.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article



The qualitative outcomes show that there are a number of impacts:

1. **Accessibility Gaps:** The AI engine also resolved more than 80% of the compliance known risks with manual efforts [11].
2. **Code Maintenance:** The development teams took considerable steps in saving their time on handling a review of codes and testing of accessibility [12].
3. **Developer Awareness:** The AI assisted developers in acquiring knowledge of accessibility rules by explaining them according to the context in the long-term [13].
4. **User Experience:** On preliminary usability testing, enhanced navigation and readability was proposed to the users who depended on assistive technologies [14].

The research has certain limitations also. The AI sometimes had problems with dynamic contents or duration of event handlers that altered the contents of the DOM at run time [15]. It also had difficulties of interpreting brand specific color schemes to compare colors. These problems highlight the ongoing relevance of the human element to sophisticated design settings in the current world.

Future studies can examine the idea of integrating NLP with reinforcement learning such that the AI will be less affected by the various structures of healthcare portals. Post-deployment validation of the solution can be further automated with integration to RPA or AIOps systems to ensure that accessibility enhancements would not be lost when updating this solution.

The findings confirm that AI uncertainty refactoring is a way to go in modernizing old healthcare. It has technical as well as social advantages to it- enhancing compliance and making digital healthcare more inclusive and sustainable.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

V. CONCLUSION

The researchers consider that the AI-enhanced refactoring provides a valuable option to enhance the accessibility of the legacy portals in healthcare. The NLP-based model was able to identify and fix majority of ARIA and semantic errors, as well as codes were functional. The AI engine cut the time spent on manual remediation though human review was still required on complicated design aspects. The conclusions prove that AI is capable of aiding this improvement of continuous accessibility on the enterprise level. The model needs to be increased by expanding on future research to accommodate the dynamic interfaces, as well as combining it with automated testing pipelines to achieve sustainable accessibility compliance with healthcare systems.

References

[1] Ferreira, F., Borges, H. S., & Valente, M. T. (2024). Refactoring react-based Web apps. Journal of Systems and Software, 215, 112105. https://doi.org/10.1016/j.jss.2024.112105
[2] Nguyen, T. (2024). Evaluating automated accessibility checker tools. In WWU Honors College Senior Projects [Thesis]. https://cedar.wwu.edu/wwu_honors/805
[3] Kapula, K. & National Science Foundation. (2024). AUTOMATED EVALUATION OF DIGITAL ACCESSIBILITY: A TEASER ON WCAG ANALYZER SYSTEMS. Journal of Computational Analysis and Applications, 2895–2902. https://doi.org/10.48047/jocaaa.2024.33.06.116
[4] Mirchev, M., Costea, A., Singh, A. K., & Roychoudhury, A. (2024). Assured automatic programming via large language models. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2410.18494
[5] Wang, Y., Wang, W., Joty, S. R., & Hoi, S. C. H. (2021). CodeT5: Identifier-Aware Unified Pre-Trained Encoder-Decoder models for code understanding and generation. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2109.00859
[6] Pandi, S. B., Binta, S. A., & Kaushal, S. (2023). Artificial intelligence for technical debt management in software development. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2306.10194
[7] Singh, P. (2024, July 3). Transforming Healthcare through AI: Enhancing Patient Outcomes and Bridging Accessibility Gaps. Journal of Artificial Intelligence Research. https://thesciencebrigade.com/JAIR/article/view/237
[8] Chaturvedi, R. (2023, May 31). *Robotic Process Automation (RPA) in Healthcare: Transforming revenue cycle operations*. https://ijritcc.org/index.php/ijritcc/article/view/11045
[9] Poghosyan, A., Harutyunyan, A., Davtyan, E., Petrosyan, K., & Baloian, N. (2024). A Study on Automated Problem Troubleshooting in Cloud Environments with Rule Induction and Verification. Applied Sciences, 14(3), 1047. https://doi.org/10.3390/app14031047
[10] Liu, M., Wang, J., Lin, T., Ma, Q., Fang, Z., & Wu, Y. (2024). An empirical study of the code generation of Safety-Critical Software using LLMS. *Applied Sciences*, *14*(3), 1046. https://doi.org/10.3390/app14031046
[11] Nie, L., Liu, H., Sun, J., Said, K. S., Hong, S., Xue, L., Wei, Z., Zhao, Y., & Li, M. (2024). SoK: Detection and Repair of Accessibility Issues. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2411.19727
[12] Vijayvergiya, M., Salawa, M., Budiselić, I., Zheng, D., Lamblin, P., Ivanković, M., Carin, J., Lewko, M., Andonov, J., Petrović, G., Tarlow, D., Maniatis, P., & Just, R. (2024). AI-Assisted Assessment of Coding Practices in Modern Code Review. *AIware 2024: Proceedings of the 1st ACM International Conference on AI-Powered Software*, 85–93. https://doi.org/10.1145/3664646.3665664
[13] Jin, H., Lee, S., Shin, H., & Kim, J. (2024). Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education. *CHI '24: Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 1–28. https://doi.org/10.1145/3613904.3642349

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

[14] Souza, A., De Souza Filho, J. C., Bezerra, C., Alves, V. A., Lima, L., Marques, A. B., & Monteiro, I. T. (2024). Accessibility Evaluation of Web Systems for People with Visual Impairments: Findings from a Literature Survey. *IHC '24: Proceedings of the XXIII Brazilian Symposium on Human Factors in Computing Systems*, 1–13. https://doi.org/10.1145/3702038.3702090

[15] Sung, C., Kusano, M., Sinha, N., & Wang, C. (2016). Static DOM event dependency analysis for testing web applications. *FSE 2016: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 447–459. https://doi.org/10.1145/2950290.2950292