https://doi.org/10.23106/dcmi.952636183

DCPAPERS

DCMI 2026

Proceedings of the International Conference on Dublin Core and Metadata Applications 2026

ISSN: 1939-1366

From Six Years to Six Hours: Prerequisites and Implications of AI-Assisted Development in a Library Context

May Chan [1]

[1] University of Toronto, Toronto, Ontario, Canada

Abstract

This paper presents a case study of using AI-assisted development to redevelop iSearch, a legacy cataloging application used daily at the University of Toronto Libraries. Redevelopment has been deferred for six years due to competing institutional priorities. Emboldened by success using AI to develop Model Context Protocol (MCP) servers, the author produced a web-based prototype replacement in approximately six hours. This paper identifies the prerequisites that enabled rapid AI-assisted development and examines emerging implications for library practitioners considering using AI similarly. The paper argues that while AI-assisted development lowers the barrier to writing code, it does not reduce the need for preparation; rather, it changes what skills and knowledge practitioners must bring to the work, and what institutions must invest in to support them.

1. Introduction

iSearch is a locally developed desktop application critical to the workflows of the Metadata Services department at the University of Toronto Libraries (UTL). It supports two core workflows: bibliographic record harvesting, in which the best MARC 21 record is selected from results returned by querying Z39.50 databases; and inventory creation, in which holdings and item-level data is generated for import into UTL's library management system (LMS). iSearch was developed by a librarian who retired without transferring the source code to the University. It cannot be updated, patched, or recovered if it fails. As such, functional requirements and record assessment criteria were documented between 2019 and 2020 in anticipation of a redevelopment project with UTL Information Technology Services (ITS). The project has been deferred by competing institutional priorities, leaving iSearch as an unresolved institutional risk.

In March 2026, the author used Claude to develop several Model Context Protocol (MCP) servers, including one connecting AI agents to Library of Congress linked open data vocabularies [1]. That work built familiarity with AI-assisted development that proved transferable. Emboldened by that experience, the author produced a web-based prototype replacement in approximately six hours. This paper reflects on what made that possible and implications for library practitioners considering AI-assisted development.

2. Prerequisites for AI-Assisted Development

The six hours of AI-assisted development did not happen in a vacuum. Several conditions enabled rapid, directed development.

Project Report

Published Sep 10, 2026

Correspondence to
May Chan
ms.chan@utoronto.ca

OPEN ACCESS

Copyright © 2026 Chan. This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International license, which enables reusers to distribute, remix, adapt, and build upon the material in any medium or format, so long as attribution is given to the creator.

2.1. Prior Documentation

Functional requirements and record assessment criteria had been written in 2019–2020 in anticipation of a collaboration with ITS to redevelop iSearch. A project proposal was prepared in March 2026 for new ITS leadership, who had signalled a willingness to begin redevelopment later in the year. This documentation gave Claude a precise structure to build toward. Rather than describing a vague need, the author could provide specific inputs such as accepted identifier types, Z39.50 query behaviour, the record selection algorithm, assessment rubric levels, SFTP delivery requirements, and LMS integration details.

2.2. Domain Expertise

Domain expertise played a role at two distinct stages of development. First, it was essential in documenting the functional requirements and record assessment criteria. Understanding the MARC 21 Formats for Bibliographic and Holdings Data [2], [3], cataloging workflows, the Z39.50 protocol, and how bibliographic records should be evaluated required specialist knowledge. The record assessment rubric is grounded in the BIBCO Standard Record (BSR) [4], a metadata application profile for bibliographic records deemed of sufficient quality by the Program for Cooperative Cataloging. Second, that same expertise was indispensable during testing. Domain knowledge was the lens through which output was evaluated at every stage.

2.3. Computational and Technical Literacies

Computational and technical literacies enable practitioners to prompt effectively, decompose a problem into manageable tasks, and recognize when generated code is superficially correct but wrong. Building MCP servers and subsequently the prototype gave the author the opportunity to apply and extend that grounding in the context of AI-assisted development. These literacies were built incrementally over the author's career, notably through The Carpentries, which provides accessible technical skill instruction through an inclusive pedagogy [5].

2.4. Access to Tools

Not everyone has equal access to the tools AI-assisted development requires. Library practitioners in environments with restrictive device management policies or limited software licensing may face significant barriers before a single line of code is written. The author works on a MacBook Air unencumbered by institutional IT policy, making it straightforward to install packages, run local servers, and experiment freely. Earlier success with MCP server development was enabled by access to Claude Edu, configured to support local connectors, without which that foundational work would not have been possible.

2.5. Critical Evaluation of AI Recommendations

Knowing how to evaluate AI recommendations critically is a distinct prerequisite. When AI tools suggest installing software or running commands, a practitioner needs enough background knowledge to verify those recommendations before acting. Vetting an

unfamiliar tool or command draws on computational and technical literacies, as well as practical familiarity with the development ecosystem. Following AI recommendations uncritically could introduce security risks or unintended dependencies.

3. The Development Process

The starting point for redevelopment was the functional requirements and record assessment criteria documented in 2019–2020, supplemented by access to the running iSearch application. A critical piece of guidance came from the retiring developer himself, who pointed to YAZ, Index Data's open-source Z39.50 toolkit [6], as the appropriate library for this work. The idea of using a tournament-based record selection algorithm was similarly informed by the developer.

Rather than building the entire application at once, the author decomposed the development process into discrete tasks, each independently testable. Claude was directed to build a Python/FastAPI backend with a static HTML/CSS/JavaScript frontend. Core components were developed sequentially: Z39.50 querying, the tournament-based record selection algorithm, record assessment against a three-level quality rubric, and the inventory creation workflow. The prototype was tested end-to-end with the exception of SFTP delivery to the production transfer server, which requires network access controlled by ITS.

While Claude was a capable collaborator for translating well-specified requirements into working code, the process was not without friction; human judgment was essential throughout. The output was sometimes subtly wrong in ways only domain expertise could detect: incorrect handling of MARC encoding levels, edge cases in record comparison logic, or assessment criteria applied in the wrong order. Testing was a continuous process, with each component verified against real Z39.50 targets and live bibliographic data.

4. Implications

The iSearch prototype demonstrated that AI-assisted development can produce working, useful software in a compressed timeframe, but building it surfaces implications for institutions seeking to create the conditions for this kind of work to succeed.

4.1. Practitioner Agency

One significant benefit of AI-assisted development was the degree of agency it afforded. In a conventional development model, a domain expert is removed from the code. Functional requirements are translated into code by a developer, the code is tested, and results are cycled back to the domain expert for review. AI-assisted development collapsed that feedback loop. The author could test against real cataloguing scenarios immediately, identify problems, and correct them in the same session. The result was a tighter iteration cycle in which the domain expert's judgment was applied directly to the code rather than mediated through a third party.

4.2. The "Vibe Coding" Risk

The term "vibe coding" describes AI-assisted development in which non-developers accept generated code without fully understanding it. The risk is significant: a practitioner without sufficient technical literacy to read and evaluate generated code might miss logical errors or introduce malicious dependencies. The prerequisites described in this paper are not optional enhancements; they are safeguards.

4.3. Coding Standards and Maintainability

AI-generated code is not automatically clean, consistent, or maintainable. Code produced by non-developers through AI-assisted development may require significant remediation before it is ready for production. Conducting a preliminary AI-assisted review against standards such as PEP 8 [7] before handing code to developers can help surface obvious issues and reduce remediation effort downstream. The author took this step ahead of planned ITS collaboration, through which formal code review is anticipated.

4.4. Accessibility

Web applications developed with AI assistance may not meet accessibility standards by default. Explicitly specifying accessibility requirements from the outset and conducting a preliminary review against WCAG 2.1 AA guidelines [8] can help surface obvious gaps before the code reaches developers. The author used AI for this review before engaging ITS; accessibility testing remains forthcoming.

4.5. Institutional IT Involvement

The iSearch prototype was developed as a proof of concept to support collaboration with ITS, not to bypass institutional processes. However, a prototype developed outside regular channels carries shadow IT risk if used in production without appropriate security review, authentication, and infrastructure support. In the iSearch case, the initial reaction from ITS was positive: the prototype was seen as saving development time. Formal redevelopment has been scheduled for late summer 2026.

5. Conclusion

This case study demonstrates that AI-assisted development can be legitimate and productive for library practitioners under certain conditions. The prototype was enabled by prior documentation, domain expertise, deliberately developed literacies, and proactive pursuit of appropriate tools. None of these prerequisites should be taken for granted; institutions envisioning AI-assisted work must invest deliberately in building them.

The experience also points toward a constructive model for how prototype development can relate to institutional IT processes. Practitioners can frame prototypes as proof-of-concept contributions that accelerate and inform formal redevelopment. As AI-assisted development becomes more accessible, the library community will need to develop shared best practices around coding standards, accessibility, security, and institutional governance; this paper offers one early data point in that conversation.

Acknowledgments

The author gratefully acknowledges the AI Kitchen, co-sponsored by the Office of the Vice-Provost, Digital Strategies and Information Technology Services, University of Toronto, for providing access to Claude Edu, which enabled the development work described in this paper.

AI Statement

This paper was drafted and revised with Claude (Anthropic, `claude-sonnet-4-6`), drawing on supplied project documentation and ongoing chat history to provide contextually informed assistance. The author takes full responsibility for the accuracy of all claims.

References

[1] M. Chan, "LC Vocabularies MCP Server." [Online]. Available: https://github.com/msuicaut/lc-vocabularies-mcp

[2] L. o. C. Network Development and MARC Standards Office, "MARC 21 Format for Bibliographic Data." [Online]. Available: https://www.loc.gov/marc/bibliographic/

[3] L. o. C. Network Development and MARC Standards Office, "MARC 21 Format for Holdings Data." [Online]. Available: https://www.loc.gov/marc/holdings/

[4] Program for Cooperative Cataloging, "BIBCO Standard Record (BSR) RDA Metadata Application Profile," 2024. [Online]. Available: https://www.loc.gov/aba/pcc/bibco/documents/PCC-RDA-BSR.pdf

[5] The Carpentries, "The Carpentries." [Online]. Available: https://carpentries.org/

[6] Index Data, "YAZ: Z39.50 Toolkit for C." [Online]. Available: https://github.com/indexdata/yaz

[7] G. van Rossum, B. Warsaw, and A. Coghlan, "PEP 8 — Style Guide for Python Code." [Online]. Available: https://peps.python.org/pep-0008/

[8] A. Kirkpatrick, J. O'Connor, A. Campbell, and M. Cooper, "Web Content Accessibility Guidelines (WCAG) 2.1," World Wide Web Consortium (W3C), 2018. [Online]. Available: https://www.w3.org/TR/WCAG21/