GLL
Geomatics, Landmanagement and Landscape No. 1 • 2026, 45–59

ISSN 2300-1496 https://doi.org/10.15576/GLL/218109

Research paper

Received: 04.02.2026
Accepted: 13.02.2026
Published: 31.03.2026

Generating a web map of digital cultural heritage using large language models: A case study of Krakow

Karol Król 0000-0003-0534-8471

Department of Land Management and Landscape Architecture,
University of Agriculture in Krakow
✉ Corresponding author: k.krol@urk.edu.pl

Summary

This article presents a case study examining the use of large language models (LLMs) in the process of creating an interactive map of Krakow's digital cultural heritage. The aim of the study was to analyse the potential of LLMs as tools supporting fast prototyping of web maps in a popularisation and narrative context. The research methodology consisted of comparing two variants of the generated map using differently configured language models, while maintaining the same point dataset and the Leaflet.js programming library. In both variants, the LLM served as a client-side application code generator, and a tool supporting the integration of descriptive data and user interface design. The results suggest that a more elaborate user interface was generated using the standard ChatGPT interface, whilst the specialised GPT model produced a solution with lower logical complexity, understood as the range of decision-making rules and data processing implemented on the client side of the application. This confirms that the degree of language model specialisation does not translate directly into the functional scope of the generated map application. The conclusions highlight the role of LLMs as tools that support the design process and shorten the prototyping cycle in web cartography, while maintaining expert control over the data and their interpretation.

Keywords

web map quality • geoinformation • geoinformatics • web cartography • large language models (LLMs) • narrative maps • digital cultural heritage • web map prototyping

1. Introduction

The rapid development of large language models (LLMs) in recent years has significantly expanded their scope of application beyond traditional natural language process-

ing (NLP) tasks. Increasingly, these models are being used as tools to support software development, user interface design and the integration of data with varying degrees of structuring. One domain where the potential of LLMs remains widely unrecognized is web cartography, particularly the ability to create interactive maps for popular science and educational purposes quickly.

In cultural heritage projects, particularly those involving digital and technological heritage, maps serve both as tools for locating sites and as vehicles for storytelling [Król 2024]. They organise disjointed information, situate them in a spatial context, and share content, which often does not qualify as formal geographic data, with a wide audience. Typical resources used in such initiatives include point locations, short textual descriptions, memoirs, or popular-science level information. These data often possess great interpretive and educational value [Dziatkiewicz and Król 2026]. Creating interactive web maps in this context involves a number of challenges. On the one hand, it requires cartographic and geoinformatics competences, and on the other, knowledge of web technologies such as HTML, CSS and JavaScript, as well as map libraries such as Leaflet [Horbiński and Lorek 2022]. In practice, this means that various roles must be combined: spatial data analyst, programmer and user interface designer, and often also a tester or junior tester [Stray et al. 2022]. For small research teams, local government bodies or grassroots initiatives, this barrier can be a significant constraint, forcing them to give up on more advanced forms of spatial presentation. This raises the question of whether LLMs can be used as tools to support the process of creating web maps, particularly during the prototyping and integration of the spatial, narrative and user interface layers.

Unlike classic desktop GIS systems, LLMs do not generate or verify spatial data. However, they can contribute to the process of data integration, map code generation, user interface design, and the interpretation of textual descriptions, which are subsequently employed in the visual layer of the map [Tao and Xu 2023]. This approach treats LLM as a component supporting the expert's work during the prototyping stage and the promotion of the results of mapping a given phenomenon. The aim of this paper is to present a case study involving the creation of an interactive map of Krakow's digital cultural heritage using large language models. The paper presents two variants of a web map generated using LLMs, which differ in their configuration and method of map generation. In both cases, the LLM served as a generator of the map application code and a tool that supported the organisation of descriptive data and the design of the user interface. The approaches are presented as an application-based case study, the aim of which is to illustrate various styles and strategies for generating map applications using LLMs and to highlight their design implications in the context of web cartography and cultural heritage projects. The paper does not attempt to provide an objective comparison of the quality of language models. The rest of the paper includes an overview of the evolution of web mapping, a description of the case study used methodology, a presentation of the obtained results, and a discussion and summary of these results.

2. Background

Over the last few decades, web mapping has evolved from static 'maps on a website' to sophisticated geoinformation applications. Historically, we can distinguish a clear chronology of 'epochs' in the development of web mapping, defined primarily by changes in information architecture, the model of user interaction with the map, and the available techniques for rendering and distributing spatial data, as well as in network bandwidth [Król 2020]. This overview allows us to situate contemporary practices, including map generation using AI, within the continuity of technical solutions that have gradually lowered the entry threshold for web map creators and increased their accessibility to users.

Solutions with limited interactivity predominated in the Web 1.0 period. Maps were mainly published as raster images (e.g. GIF/JPEG) embedded within the structure of an HTML hypertext document, possibly with simple navigation mechanisms that usually consisted of hyperlinks to subsequent raster compilations. The application logic was usually implemented on the server side, and the user received successive views in response to HTTP requests [Król 2020]. In practice, this meant a high cost of 'refreshing' the map and limited fluidity of exploration. The map was primarily designed for 'reading and browsing', and the configuration of its view was usually predesigned. At the same time, map service standards (e.g. WMS) were being developed, enabling the publication of maps as web services, which gradually streamlined the server-side layer of Web GIS [Michaelis and Ames 2008].

Another milestone in online mapping was the rise of an approach typical of the Web 2.0 period, in which a key role was played by: 1) asynchronous communication between the browser and the server (AJAX), 2) the 'slippy map' interface with continuous panning and zooming, and 3) the widespread adoption of public map APIs and mashup practices. It is emphasized in research literature that the use of AJAX in mapping services (led by Google Maps) enabled continuous navigation across map tiles loaded on demand, which transformed users' expectations regarding the interactivity of online maps [Roth et al. 2014]. In the same period, the concepts of 'GeoWeb', participatory mapping and neogeography began to be systematically described. Mapping became a widespread social practice, fuelled by content created by users rather than solely by cartographic institutions [Haklay et al. 2008].

From a technical perspective, modern web maps have been dominated by two key features: the Web Mercator projection and raster tile maps (XYZ/tiles). This has enabled the efficient distribution of map data and the scaling of services to a large number of users, whilst maintaining a consistent navigation model. It is argued that the combination of Web Mercator and raster tiles has become the basis for modern map service providers, determining how maps are presented online and the ecosystem of developer tools [Stefanakis 2017]. Over time, mature web mapping architectures began to integrate various data sources (services and files), whilst the development of browsers and web standards (HTML5, CSS3, JavaScript) shifted a significant portion of the logic of application to the client-side. In the next phase of development (often associated with the 'Web 3.0+'), there has been a growing emphasis on vector data, data exchange formats (particularly

GeoJSON), JavaScript libraries, and hardware-accelerated rendering (WebGL) [Zunino et al. 2020]. Lightweight map libraries, such as Leaflet, have become particularly important, as they enable the creation of interactive maps without the need for advanced GIS infrastructure on the user's side [Cegielska et al. 2018]. Studies presenting workflows based on Leaflet point to its usefulness in creating map applications that combine a spatial layer with multimedia content or narrative, whilst maintaining a relatively low level of implementation complexity [Edler and Vetter 2019]. Schemes for integrating historical data with a web map through conversion to GeoJSON and implementation in Leaflet show that modern web mapping tools support not only reference maps but also thematic maps, including reconstructions and archival presentations [Horbiński and Lorek 2022]. Against this backdrop, LLM can be interpreted as a further 'accelerator' of the work of web map creators: it does not replace rendering mechanisms (tiles, Web Mercator, vectors), but can support the generation layer of the application (code generation), the semantic layer (organising descriptions, classifying objects for filters/legends) and the interface layer (design of panels, search functions, map behaviour).

Large Language Models (LLMs) represent a class of artificial intelligence systems developed within the field of Natural Language Processing (NLP), whose operation relies on machine learning applied to very large text datasets. These models are trained to predict the next tokens in a sequence, enabling them to generate coherent text, analyse linguistic content and perform tasks such as summarisation, classification, information extraction and code generation. LLM architectures, including models from the GPT (Generative Pre-trained Transformer) family, are based on the transformer mechanism and pre-training, and can subsequently be fine-tuned for specific applications. In the context of artificial intelligence in general, LLMs are not reasoning systems or tools for semantic analysis in the symbolic sense, but probabilistic language models whose usefulness stems from their ability to integrate linguistic, contextual and structural knowledge. The scientific literature emphasises that their role lies primarily in supporting creative and design processes by generating textual and programming artefacts, rather than in autonomous reasoning or data verification [Smajić et al. 2025]. From the perspective of the history of web mapping, this is a continuation of the trend towards lowering competence barriers and shortening the prototyping cycle, observed since the shift from static maps to 'slippy map' applications and JavaScript libraries. The interpretative framework of 'periods' in web mapping helps to avoid overestimating the role of new tools. LLM operates within a mature ecosystem of web technologies, and its value is revealed primarily in the integration of code, data and narrative within a short development cycle [Tao and Xu 2023].

3. Materials and methods

The case study focused on an interactive map of Krakow's digital cultural heritage, featuring the locations of institutions, venues and organisations involved in the fields of electronics, digital technologies and video game culture. The input dataset was heterogeneous and poorly structured, i.e. it comprised point locations and textual descrip-

tions of varying form and content, without a uniform attribute schema or formal classification of objects [Dziatkiewicz and Król 2026]. After structuring, each data record included geographical coordinates (latitude and longitude), the name of the object, and a short informative or popular-scientific text description. The data did not contain any explicit temporal attributes or a formal classification of object types.

The source of the data included the author's own studies, project materials and publicly available descriptive information, compiled for initiatives related to the mapping of Krakow's digital cultural heritage [Dziatkiewicz and Król 2026]. The data were point-based and were used solely to demonstrate the process of creating a web map. They were not subject to statistical or spatial analysis in a quantitative sense. Responsibility for the factual and locational validity of the data remained with the author. The LLM model was not used to verify the data or generate coordinates.

3.1. The use of LLM in the process of map creation

This paper presents two versions of an online map generated using an LLM. In both cases, the model was employed as an application code generator and as a tool for supporting the semantic organisation of descriptive data and the design of the user interface.

3.1.1. ChatGPT as a web map code generator

In the first variant, the map was generated using ChatGPT (GPT-5.2). The model's role was limited to assisting with the generation of client-side code (HTML, CSS and JavaScript), without being used to generate, modify or verify spatial data. ChatGPT did not function as a runtime environment or a GIS system, but worked as a programming assistant, supporting rapid prototyping of a web map based on a textual description.

The map-generation process was iterative and interactive. The user formulated successive instructions regarding the application's structure, the expected user interface functions and the method of presenting point data, whilst the model generated the corresponding code fragments. This process resulted in a complete, single-file HTML document that integrated the cartographic layer, user interface elements and interaction logic. The generated application was based on the Leaflet.js library and publicly available OpenStreetMap tiles, and required no additional technological dependencies or server infrastructure.

ChatGPT was mainly used to generate the structure of the map application, including map initialisation, point marker management and implementation of user interaction mechanisms. As part of this approach, the model also provided support for the implementation of user interface features, such as an information panel, object filtering, text search and automatic adjustment of the map view to the currently displayed data. The scope and form of these features were the result of a dialogue with the user (CUI) and could be modified in subsequent iterations of code generation.

It should be stressed that ChatGPT did not make any autonomous design or cartographic decisions. Responsibility for selecting the map library, the data structure, the

symbolisation of features, and the functional scope of the application remained with the user. The model served as a tool to support code integration and accelerate the map creation process, rather than being an independent system for designing a mapping application.

The use of ChatGPT in this role accelerated the generation of a working prototype of a web map, which could then be manually edited and adapted to the needs of the case study. This approach aligns with the practice of using LLMs as tools to support software development, rather than as autonomous generators of solutions with guaranteed quality. For the purposes of this study, ChatGPT was treated as an element of the design process, the effects of which were evaluated on the basis of the generated technical artefact, presented further on in this paper.

3.1.2. The GPT model as a web map code generator

StoryMap Web Builder (SMWB) is a specialised custom GPT conversational model, built in the ChatGPT ecosystem, designed to automatically generate single-file, narrative web maps (story maps) in the HTML format. The tool functions as a conversational user interface (CUI): the user provides point data or input files (e.g. KML), and the model generates a complete, ready-to-use HTML5 document, containing the structure, style (CSS) and application logic (JavaScript). The SMWB specification is declarative in nature (Table 1) – not all of the listed features need to be present in the maps generated for a specific case study.

The mapping layer is based on the Leaflet.js library (v1.9.x), loaded from a public CDN (Content Delivery Network). OpenStreetMap tiles are used as the map background, while all other functionality works offline. The tool does not use API keys or external web services. Spatial data (points) are embedded directly in the JavaScript code as JSON objects. The UI design is responsive (mobile-first), high-contrast and compliant with best practices for accessibility.

Table 1. Declared (design) technical specifications for the StoryMap Web Builder tool*

Component	Specification
System type	Conversational model (custom GPT)
Platform	ChatGPT, OpenAI
Language model	A GPT-class language model (ChatGPT, OpenAI)
Mode of operation	Generative, dialogue-based (text → HTML code)
Task	Automatic generation of narrative web maps
System output	Complete single-file HTML5
Web standards	HTML5, CSS3, JavaScript (ECMAScript 2020+)
Map library	Leaflet.js v1.9.x (CDN)

Map basic data	OpenStreetMap tiles
External dependencies	Lack (except for OSM tiles)
Frontend frameworks	Not used (no React/Vue/Angular)
Supported input data	CSV, GeoJSON, KML, textual data
Data model	Points (lat, lon, descriptive attributes)
Handling missing data	Signalling + user decision
Generating descriptions	2–3 sentences, marked as 'editorial description'
Security	Text escaping (escapeHtml()), no eval
Information panel	Collapsible, open by default
Adjusting the view	Automatic (fitBounds)
Interface style	Neutral, typical of museum and education apps
Responsivity	Mobile-friendly (RWD)
Offline mode	Yes (except for map tiles)
Runtime environment	Browsers (Chrome, Firefox, Safari, Edge)
Model management	On the OpenAI side (hosting, inference)
Replicability	High (static HTML, no backend)

* StoryMap Web Builder is treated in this study as a black-box generative assistant; the object of comparison is the generated HTML artifact rather than the internal model logic.

StoryMap Web Builder supports input data in CSV, GeoJSON and KML formats, and performs automatic point extraction. Should there be any gaps in the narrative data, the system alerts the user and asks whether the descriptions should be completed manually or generated automatically (2–3 sentences, labelled as 'editorial description'). All text fields are protected by the escapeHtml() function to prevent XSS vulnerabilities.

The tool operates in standard browser environments, with no server requirements, which promotes its archival durability and the replicability of research. It should be pointed out, however, that the specification presented refers to the tool's declared and designed functions. The functional scope of the generated maps in a given case study may vary depending on the model configuration and the course of the conversation with the user.

4. Results

The use of LLMs in the process of creating an online map led to the generation of two fully functional versions of an interactive map of Krakow's digital cultural heritage, produced as single-file web applications (HTML). However, these variants differed in

terms of the complexity and integration of the user interface, despite using the same map library (Leaflet.js) and the same set of input data.

The map-generation process was iterative and interactive. Below we present an example of the types of instructions (prompts) that were used when working with language models and that led to the creation of the presented artefacts (Table 2).

Table 2. Reconstruction of sample prompts used in two variants of generating a web map using an LLM

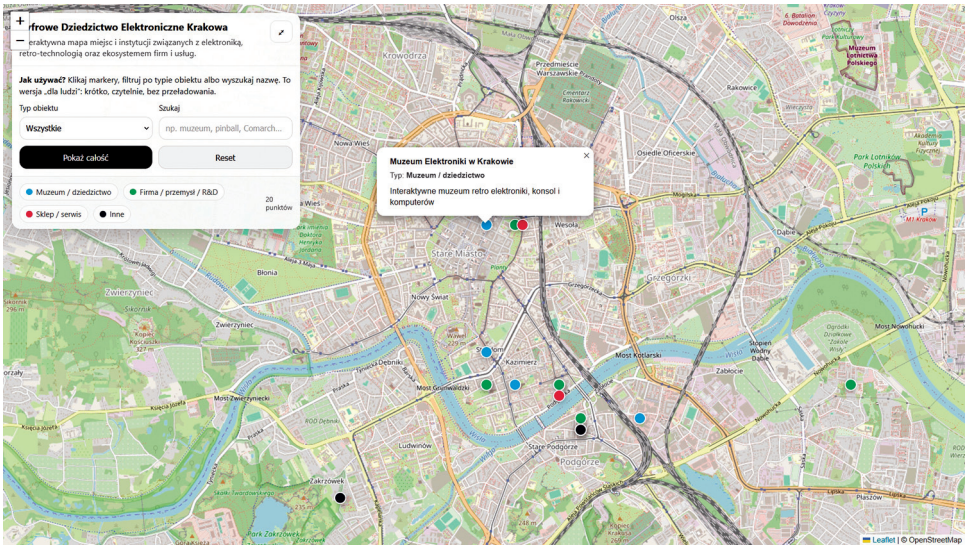
Prompt – standard version ChatGPT
Prompt 1 (reconstructed): Create a single-file HTML web map using Leaflet.js and OpenStreetMap tiles. The map should present point locations of digital cultural heritage in Kraków (institutions, places, initiatives related to electronics and video game culture). Use client-side HTML, CSS and JavaScript only, without any backend or API keys. Add an interactive user interface including: a collapsible side panel with project description, category-based filtering of points, text search across names and descriptions, a dynamically generated legend, automatic adjustment of map extent to visible points. Ensure the application is self-contained, readable, and easy to modify.
Prompt – specialised GPT model version (StoryMap Web Builder)
Prompt 2 (reconstructed): Generate a narrative web map as a single HTML file using Leaflet.js and OpenStreetMap tiles. The map should present a list of point locations related to digital cultural heritage in Kraków. Use a simple, stable layout with: a fixed side panel listing all objects, click-to-center interaction between the list and the map, basic pop-ups with object descriptions. Do not include filters, search, or dynamically generated legends. Focus on clarity, determinism, and ease of maintenance.

The reconstruction of the prompts indicates that the process of generating online maps using an LLM was iterative. In the case of the standard ChatGPT interface, the instructions led to a gradual expansion of the map application's functionality and the development of the user interface. In contrast, in the version using a specialised GPT model, a declarative style prevailed, focused on simplicity and stability of the solution, which resulted in a reduction in the number of interface elements and logical rules.

4.1. Map generated using the ChatGPT interface

The first version of the map, generated using the standard ChatGPT interface, was an extensive client-side narrative application, designed as a single-file web application in HTML5. The generated document integrated the document structure (HTML), the

presentation layer (CSS) and the application logic (JavaScript) into a single file, without the need for front-end frameworks, compilation processes or server-side infrastructure. The cartographic layer was based on the Leaflet.js library (v1.9.x), and OpenStreetMap tiles loaded from a public server were used as the map base (Fig. 1).



Source: Author's own study

Fig. 1. An interactive map generated using ChatGPT

The user interface featured a collapsible information panel, which provided the main point of interaction with the map. This panel included a project description, filter controls and navigation elements, and its collapsibility allowed the map area to be maximised during exploration. Mechanisms were implemented to filter objects by thematic categories, with the list of available categories generated dynamically based on input data, without the need for manual definition of class dictionaries. This solution made it possible to automatically adapt the interface to changes in the dataset.

The interface also included an additional feature – a text search engine that allowed users to search through the names and descriptions of objects. The search mechanism operated on the client side and was implemented by comparing character strings within data embedded in JavaScript code. The search could be combined with category filtering, enabling multi-criteria exploration of spatial data. The results of filtering and searching were reflected on the map by updating the visible markers.

A key feature of the application was a dynamic legend, generated automatically based on the categories assigned to the objects. Each category was assigned a distinct marker colour, which made it easier to visually distinguish between different types of objects on the map. A counter for visible points was also implemented, which updated

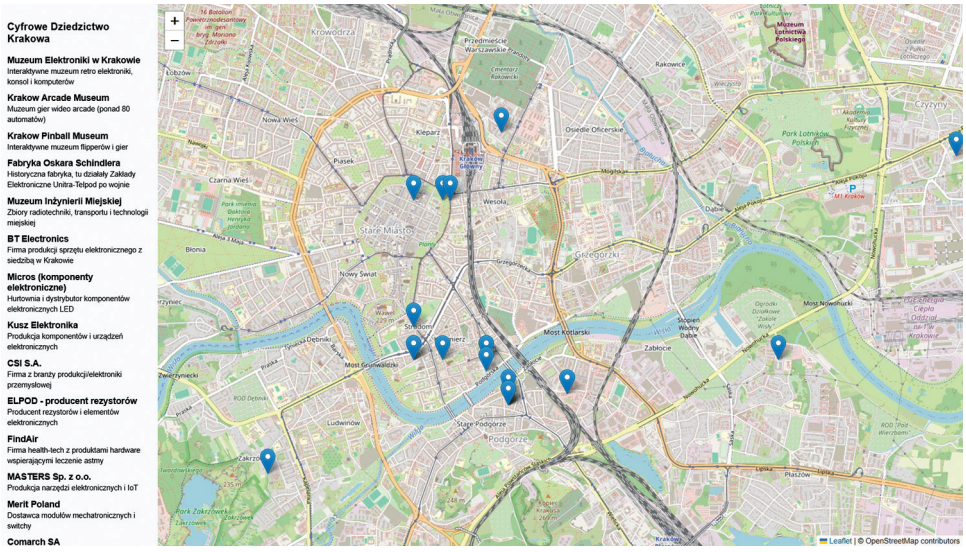
after every change to the filters or search query, providing the user with feedback on the current scope of the displayed data.

The map's logic included mechanisms for automatically adjusting the view to the currently displayed objects (the fitBounds function), which allowed users to quickly focus on a selected subset of data without manually zooming in or panning the map. The user also had the option to reset the filters and restore a view showing all objects. All interactions were executed on the client side, providing a smooth user experience and ensuring the application's independence from the server infrastructure.

From a technical point of view, the map that was generated with the standard ChatGPT interface constituted a complete, functional prototype of a narrative web map, combining a spatial layer with mechanisms for selecting and exploring content. This solution was relatively easy to modify and adapt further, and its code structure remained readable and transparent to users with a basic knowledge of web technologies.

4.2. Map generated using StoryMap Web Builder

The second version of the map (Fig. 2) was generated using a specialised GPT model – StoryMap Web Builder. The result was a single-file web application with a simplified, clear user interface structure. The generated HTML document integrated the map layer, side panel and interaction logic in a minimalist form, limiting the number of UI components to those directly supporting basic exploration of point data. As in the version described in section 4.1, the cartographic layer was based on the Leaflet.js library (v1.9.x), and OpenStreetMap tiles were adopted as the map base.



Source: Author's own study

Fig. 2. Interactive map generated using StoryMap Web Builder (GPT model)

The main feature of the interface was a fixed side panel with a list of all the objects included in the dataset. This list was directly synchronised with the map layer: clicking on a selected object in the panel automatically centred the map view on the object's location and opened a pop-up window displaying basic descriptive information. This mechanism allowed data to be explored in 'list → map' mode, typical of simple reference and catalogue applications.

User interaction with the map was limited to basic navigation functions, such as zooming in, zooming out and panning the map, as well as viewing information about features through markers and pop-up windows. Unlike the version generated by the standard ChatGPT interface, the application did not include dynamic filters, a text search function or an automatically generated legend. All objects were displayed simultaneously, which simplified the application's logic and ensured its deterministic behaviour.

From the point of view of application architecture, the map generated by StoryMap Web Builder featured a low degree of logical complexity – understood as a limited number of decision rules and application states on the client side – owing to the absence of filtering, search and dynamic data processing mechanisms. Point data was embedded directly in the JavaScript code as an array of objects, and the handling of events was restricted to responding to user clicks. The absence of client-side filtering and data processing mechanisms reduced the risk of logical errors and facilitated further manual editing of the code by the user. Such map design functioned as a simple spatial catalogue, in which the list of objects formed an interface element on a par with the map itself. This solution can be interpreted as a stable and easy-to-maintain variant of a web map, particularly useful in situations where the goal is to quickly provide a set of locations along with descriptions, without the need for complex exploration mechanisms.

5. Discussion

A comparative analysis of the two map versions shows that the differences between the results do not stem from the used mapping technologies or the nature of the input data, but from the different ways in which the map generation process using an LLM was configured and executed. In both cases, the same mapping library (Leaflet.js) and an identical set of point data were employed, yet the final mapping applications ended up with distinctly different functional profiles.

The map generated with the standard ChatGPT interface featured a more complex user interface layer that included dynamic filters, a search function, a legend and mechanisms for automatic view adjustment (Table 3). This solution facilitates data exploration in a narrative and thematic mode, but at the same time introduces a greater number of logical dependencies on the client side, which may increase the risk of errors and make further expansion of the application more challenging. In contrast, the version generated by StoryMap Web Builder represents a minimalist approach, in which user interaction is based on a direct relationship between the list of objects and the map. The absence of filters, a search function and a legend reduces the logical complexity

and results in a high degree of determinism in the application's behaviour, understood as an unambiguous and predictable system response to user actions, independent of previous interactions and intermediate states of the interface. Such a solution is easier to maintain and edit, and its operation remains fully predictable, albeit at the cost of limiting the possibilities for selective data exploration.

Table 3. Comparison of map variations generated with the standard ChatGPT interface and StoryMap Web Builder

Feature / aspect	Standard ChatGPT interface	StoryMap Web Builder (GPT model)
Type of application	Single-file web app (HTML5)	Single-file web app (HTML5)
Generation mode	Iterative dialogue (general chat)	Custom GPT model
Map library	Leaflet.js v1.9.x	Leaflet.js v1.9.x
Map base	OpenStreetMap (kafle XYZ)	OpenStreetMap (kafle XYZ)
Architecture	Client-side, no backend	Client-side, no backend
Interface organisation	Collapsible narrative panel + map	Fixed side panel + map
Information panel	Yes (collapsible, multi-sectional)	Yes (fixed, object list)
Category filters	Yes (dynamic)	No
Text search	Yes (name + description)	No
Legend	Yes (dynamic, generated from data)	No
Counter of visible objects	Yes	No
List–map relation	Indirect (via filters and search)	Direct (click → centering + pop-up)
Adjusting the view	Automatic (fitBounds)	Manual
Complexity of logic	Average	Low
Determinism in behaviour	Average (depending on filters)	High
Code editability	High	Very high
Risk of logical errors	Moderate	Low
Application profile	Narrative / educational map	Catalogue/reference map
Entry threshold for users	Average	Low

The key insight that can be drawn from the comparative analysis is that the degree of specialisation of the GPT model does not translate linearly into the degree of functional complexity of the generated map. In the present case study, a more sophisticated

interface was generated by the general-purpose model, whilst the specialised model generated a simpler solution that was more deterministic in terms of application behaviour. This observation highlights the importance of LLM configuration and the transparency of generation rules as methodological elements (prompts), which should be documented in the same way as the selection of mapping technologies or data structures.

6. Summary

The presented case study demonstrates that large language models can successfully support the creation of interactive online maps designed for public engagement and storytelling. However, a clear division of roles must be maintained between the spatial data layer, the mapping technology, and the mechanisms for generating code and the user interface. In this study, the LLM did not function as a GIS system or a spatial analysis tool, but instead was used as a component to support the development of a web application, integrating HTML, CSS and JavaScript in a short prototyping cycle.

An analysis of two map versions generated with different LLM configurations showed that the degree of model specialisation is not a strong predictor of the quality or complexity of the output. Contrary to intuitive expectations, a more elaborate user interface was generated using the standard ChatGPT interface, whilst the specialised GPT model generated a solution characterised by lower logical complexity. This result highlights the importance of model configuration and the approach to conversation as factors that have a significant impact on the final form of the map application.

From the perspective of web cartography, the study confirms that LLMs should be viewed as tools that lower the threshold for entry and shorten the prototyping cycle, rather than as autonomous map generators offering guaranteed quality. Even with identical input data and the same map library, different model configurations can lead to varying trade-offs between interface complexity, readability, code editability and the determinism of the application's behaviour.

6.1. Practical implications and limitations of the study

The results have direct implications for the practice of creating web maps, particularly in educational and outreach projects. LLMs can be utilised as tools to support fast prototyping of web maps, enabling the integration of point data, application logic and the user interface without the need for complex GIS infrastructure or front-end frameworks. The single-file HTML applications generated in this way are easy to run, archive and modify in the future, which is important in institutional contexts with limited technical resources.

The results also indicate that the choice between a general-purpose model and a specialised model should depend on the project's objective. In situations where interface flexibility and the ability to iteratively tailor map functions to the user's preferences are important, ChatGPT's standard interface may prove more suitable. Conversely,

specialised models may prove beneficial in cases requiring simple, stable and easy-to-maintain catalogue-based solutions. From a practical point of view, the configuration of the LLM, including the transparency of prompts and the ability to iteratively refine requirements, should be treated as an element of the design process, similar to the selection of a mapping library, data schema or user interface pattern.

The presented study is an applied case study and thus does not provide a basis for statistical generalisations. The analysis covered a single dataset and a single thematic context. LLMs were not involved in the generation or verification of spatial data, and responsibility for their accuracy remained with the author. Another limitation is the lack of full transparency regarding the generation rules for specialised models. Further research could involve comparing a larger number of LLM configurations and assessing the usefulness of the maps in various application contexts.

Acknowledgments

The research was inspired by activities carried out under scientific and research mini-projects entitled: 1) Opracowanie mapy krakowskich salonów gier arcade jako części Turystycznego Szlaku Cyfrowego Dziedzictwa Kulturowego (KrADE) (Development of a map of Krakow's arcade halls as part of the Digital Cultural Heritage Tourist Trail), no. K/1786/2025/WRE-KSPS, and 2) Mapowanie muzeów cyfrowego dziedzictwa kulturowego w Polsce (DigiMap) (Mapping of digital cultural heritage museums in Poland), no. K/658/2024/WRE. Co-financed by the Minister of Science under the 'Regional Initiative of Excellence' Programme, agreement no.: RID/SP/0039/2024/01, co-financing amount: PLN 6,187,000.00, implementation period 2024–2027.

Funded by a grant from the Ministry of Science and Higher Education to the University of Agriculture in Krakow for 2026.

References

Cegielska K., Salata T., Kudas D., Szylar M. 2018. Concept of municipality geoportal – selected legal and administrative issues. Geomatics and Environmental Engineering, 12(1), 45–57. https://doi.org/10.7494/geom.2018.12.1.45

Dziatkiewicz Ł., Król K. 2026. Krakowski Szlak Arcade. Opracowanie mapy krakowskich salonów gier arcade jako części Turystycznego Szlaku Cyfrowego Dziedzictwa Kulturowego (KrADE). Kraków: DigitalHeritage. https://doi.org/10.6084/m9.figshare.31223476

Edler D., Vetter M. 2019. The Simplicity of Modern Audiovisual Web Cartography: An Example with the Open-Source JavaScript Library leaflet.js. KN J. Cartogr. Geogr. Inf., 69, 51–62. https://doi.org/10.1007/s42489-019-00006-2

Haklay M., Singleton A., Parker C. 2008. Web mapping 2.0: The neogeography of the GeoWeb. Geography Compass, 2(6), 2011–2039. https://doi.org/10.1111/j.1749-8198.2008.00167.x

Horbiński T., Lorek D. 2022. The use of Leaflet and GeoJSON files for creating the interactive web map of the preindustrial state of the natural environment. Journal of Spatial Science, 67(1), 61–77. https://doi.org/10.1080/14498596.2020.1713237

Król K. 2020. Evolution of online mapping: from Web 1.0 to Web 6.0. Geomatics, Landmanagement and Landscape, 1, 33–51. https://doi.org/10.15576/GLL/2020.1.33

Król K. 2024. Mapping digital cultural heritage museums in Poland. Scientific Papers of Silesian University of Technology – Organization and Management Series, 206, 397–417. https://doi.org/10.29119/1641-3466.2024.206.23

Michaelis C., Ames D. 2008. Web Feature Service (WFS) and Web Map Service (WMS). In: Encyclopedia of GIS. S. Shekhar, H. Xiong (eds). Springer, Boston, MA. https://doi.org/10.1007/978-0-387-35973-1_1480

Roth R.E., Donohue R.G., Sack C.M., Wallace T.R., Buckingham T.M. 2014. A process for keeping pace with evolving web mapping technologies. Cartographic Perspectives, 78, 25–52. https://doi.org/10.14714/CP78.1273

Smajić A., Karlović R., Bobanović Dasko M., Lorencin I. 2025. Large Language Models for Structured and Semi-Structured Data, Recommended Systems and Knowledge Base Engineering: A Survey of Recent Techniques and Architectures. Electronics, 14(15), 3153. https://doi.org/10.3390/electronics14153153

Stefanakis E. 2017. Web mercator and raster tile maps: two cornerstones of online map service providers. Geomatica, 71(2), 100–109. https://doi.org/10.5623/cig2017-203

Stray V., Florea R., Paruch L. 2022. Exploring human factors of the agile software tester. Software Quality Journal, 30, 455–481. DOI: 10.1007/s11219-021-09561-2

Tao R., Xu J. 2023. Mapping with chatGPT. ISPRS International Journal of Geo-Information, 12(7), 284. https://doi.org/10.3390/ijgi12070284

Zunino A., Velázquez G., Celemín J.P., Mateos C., Hirsch M., Rodriguez J.M. 2020. Evaluating the performance of three popular Web mapping libraries: A case study using Argentina's life quality index. ISPRS International Journal of Geo-Information, 9(10), 563. https://doi.org/10.3390/ijgi9100563