PREPRINT

Author-formatted, not peer-reviewed document posted on 17/03/2026

DOI: https://doi.org/10.3897/arphapreprints.e191900

Using AI-Assisted Coding to Build Digital Tools for Natural History Collections

Victor De Brito

Using AI-Assisted Coding to Build Digital Tools for Natural History Collections

Author: Victor de Brito

Cornell University Museum of Vertebrates, Department of Ecology and Evolutionary Biology, Cornell University, Ithaca, New York 14850, USA

Email: victordebrito.nf@gmail.com

Author contribution: VdB conceived the project, performed data collection, prompt development, code execution, and wrote the manuscript.

Funding: The author has no funding to report.

Competing interests: The author has declared that no competing interests exist.

Acknowledgments: The author thanks Mary Margaret Ferraro, Casey Dillman, Vanya Rohwer, and Irby Lovette for the support during the development of this project.

Data Resources: The applications and data are available on GitHub (https://github.com/debrito-victor/).

Abstract

Natural History Collections are invaluable sources of biological information for research, education, conservation, and public outreach but often lack the resources, programming expertise, and infrastructure needed to create custom digital tools for data accessibility. Large scale data aggregators facilitate data access but partially address common institutional needs. Artificial Intelligence (AI) has rapidly changed the way developers create, maintain, optimize, and test code, lowering the technical barrier to software creation. AI-assisted coding can accelerate the development of tools for digitization, data management and visualization, and public accessibility in Natural History Collections. Here I demonstrate the use of AI-assisted coding in the development of two custom web applications designed to address common institutional needs. Two web applications were built using GPT-5.3-Codex. Both were based on data exported from the Cornell University Museum of Vertebrates database. The first tool is a specimen search portal enabling multi-field filtering, data export, and mapping of localities. The second contains an interactive map to visualize the sampling of localities for four collections and a cumulative graph of specimen and lot sampled through time. Both applications performed the intended functions accurately, demonstrating that AI-assisted coding can be used to generate operational, lightweight applications for Natural History Collections. The development of web application through AI-assisted coding reduces technical barriers for collections, enabling the creation of tailored tools with minimal infrastructure. Careful prompt design, refinement, and attention to data security are essential for robust results and safe adoption of this technology.

Key-words: Application Development; Artificial Intelligence; Data Visualization; Large Language Models; Natural History Digital Tools; Programming.

Introduction

Natural History Collections are invaluable sources of biological information for research, education, conservation, and public outreach (Danks 1991, Suarez and Tsutsui 2004, Rocha et al. 2014, Rohwer et al. 2022, Davis 2024). A central goal of these collections is to make this increasing volume of data accessible, so researchers, educators, and other stakeholders can use this information directly or request access to the specimens. In most cases, data availability is achieved through institutional websites and web applications. However, limited access to programming expertise, dedicated personnel, and lack of infrastructure for large data management are major challenges for many Natural History Collections.

Large-scale initiatives have been established to increase data availability across collections, including Global Biodiversity Information Facility (GBIF), VertNet, and MorphoSource (Constable et al. 2010, Boyer et al. 2016, Anon 2026). While these initiatives have substantially facilitated information accessibility, they often don't present the most updated information from collections, which upload their data in episodic inflows. They are also subject to shifts in funding policies that may impose limits on data hosted by them or subject partner institutions to new fees. Moreover, large aggregators are typically not tailored to the specific needs of individual collections, and the acknowledgment of hosting institutions is often overlooked by authors in metadata analyses.

Recent advances in generative artificial intelligence (AI), in particular large language models designed to understand, generate, and process human-like text, have revolutionized the way developers create, maintain, optimize, and test code (Becker et al. 2023). AI-assisted coding is now widely used in software development. For instance, the rapid growth of GitHub, the largest repository for software development projects, added more than 36 million developers in 2025, reaching over 180 million users, a surge largely attributed to the adoption of advanced AI coding tools (GitHub Staff 2025). By translating natural language instructions into executable code, AI substantially lowers the technical barrier to software creation.

In Natural History Collections, AI has the potential to accelerate the development of tools for digitization, data management and visualization, and public accessibility. Web applications created through AI-assisted coding can provide accessible, low-cost tools tailored to specific institutional needs. This enables collections staff without formal training in software engineering to design and deploy digital tools.

Here, I list two prompts used to develop applications designed to address common needs in Natural History Collections based on data from the Cornell University Museum of Vertebrates: (1) a portal for specimen search and (2) an interactive map displaying sampling localities.

Methods

The two applications were developed using GPT-5.3-Codex (OpenAI 2026) on Windows 11 through Visual Studio Code, employing a medium reasoning effort. The applications were built and tested locally and later deployed online using GitHub Pages.

Specimen Search Web Portal

ARPHA Preprints Author-formatted, not peer-reviewed document posted on 17/03/2026 DOI: https://doi.org/10.3897/arphapreprints.e191900

I developed this application based on a .csv file exported from a query in the Cornell University Museum of Vertebrates database using Specify 7 v7.7.5 (Specify Collections Consortium 2022). The dataset included the following fields: order, family, genus, species, CUMV catalog number, preparation type, count, year, month, day, latitude, longitude, country, state, and county. I designed the portal to filter results based on user-defined criteria, export filtered results as a .csv file and display an interactive map showing the sampling localities of the selected specimens.

To develop the collection search portal, I created a project folder containing a .csv file with the dataset to be made searchable and an image .png file to be used as the institutional logo. All application files were generated within this directory. The following prompt was then executed in the project folder:

Prompt 1

You are a senior full-stack engineer. Build a complete, production-ready React application named for the Cornell University Museum of Vertebrates Ichthyology Collection.

Tech stack:

- React + TypeScript + Vite
- Tailwind CSS
- Leaflet (react-leaflet) for map
- PapaParse

App Requirements:

1. Main layout:
- Header at top
    - Logo: `/CUMV_logo.png` (.png file name)
    - Subtitle text: `Cornell University Museum of Vertebrates` in Times New Roman
    - Title text: `Collection Database`
- Left filter panel about 1/4 of the page
- Right content with pagination/export, results table, and map, about 3/4 of the page
- Footer with contact info centered

2. Csv-Based Search & Filtering
- Load the CSV file from the local project directory.
    - CSV file is local in project: `fish_collection.csv`
- Automatically generate searchable filter fields based on the CSV column headers.
- Each filter field must support:
    - Dropdown scroll list with all unique values available
    - Autocomplete suggestions while typing
    - Multi-select filtering (filter by more than one value)
    - Show the selected values as cards on the active fields

ARPHA Preprints Author-formatted, not peer-reviewed document posted on 17/03/2026 DOI: https://doi.org/10.3897/arphapreprints.e191900

3. Dynamic Filter Interdependence
- When the user filters by one field, update all other dropdowns so they only show values that are valid given the current filtered dataset.
    - OR logic within a field, AND logic across fields
    - Example: If "Country = Peru" is selected, the "Species" dropdown should only show species that occur in Peru.

4. Results Table + Export
- Show filtered results in a paginated table.
- Pagination options: 20 / 50 / 100 results per page.
- Allow exporting the currently filtered results table as a new CSV file.
    - Export button: exports only currently filtered dataset to CSV

5. Interactive Specimen Map Panel
- Add a panel with an interactive map.
- Plot all specimens that have latitude and longitude.
    - Plot all filtered rows with valid coordinates
    - Map updates live as filters change
- Clicking a map point should highlight/select the corresponding row in the results table.
- The map should be optimized to support a desktop and mobile version of the application

6. Theme and Styling
- Page background white
- Cards/boxes light blue
- Ensure readable dark text on light backgrounds
- Soft SaaS look
- Glassmorphism cards
- Subtle blur, rounded corners, clean typography
- Maintain clean spacing and modern layout on desktop and mobile

7. Footer information
- Cornell University Museum of Vertebrates
- 159 Sapsucker Woods Road
- Ithaca, NY 14850-1923
- (607) 255-3682
- https://www.cumv.cornell.edu/ (clickable)
- Collection Manager:
- Victor de Brito, Ph.D.
- Email: victordebrito@cornell.edu (clickable)

ARPHA Preprints Author-formatted, not peer-reviewed document posted on 17/03/2026 DOI: https://doi.org/10.3897/arphapreprints.e191900

- Curator:
- Casey B. Dillman, Ph.D.
- Email: cbd63@cornell.edu (clickable)

8. README
- Include a README.md with step-by-step instructions for someone unfamiliar with coding:
    - How to install the application
    - How to run the app locally
    - How to properly close/stop the application

Interactive Map Showing Sampling Localities Over Time

I developed an interactive map displaying sampling numbers for the four collections at the Cornell University Museum of Vertebrates. Sampling counts were based on individual specimens for the mammal and bird collections, and lots for ichthyology and herpetology. The application visualizes localities using color-coded points to distinguish collections and includes a line graph showing the cumulative number of specimens and lots over time.

Similar to the first case, the application was developed based on a .csv file exported from a query in Specify 7, but in this case, focused on locality data. The dataset included the following fields: catalog number, class, order, family, genus, species, year, country, state, county, latitude, and longitude.

A project folder was created containing the .csv dataset and an image .png file to serve as the institutional logo. All application files were generated within this directory. The following prompt was then executed in the project folder:

Prompt 2

You are an expert full-stack developer and scientific data visualization engineer. Build a complete local web app in the current project folder to visualize museum specimen localities through time from a CSV file.

Project files already present:

- cumv_localities_all.csv
- CUMV_logo.png

Create all needed app files (at minimum):

- index.html
- style.css
- main.js
- package.json
- README.md

Tech stack:

- Frontend-only app (no backend required)
- Leaflet for interactive map
- Plotly for line chart
- PapaParse for CSV parsing
- Node static server via npm start

Functional requirements:

1. Data loading and grouping
- Load data from cumv_localities_all.csv.
- Use these CSV columns:
  - Latitude1
  - Longitude1
  - Began Date (Year)
  - Class
- Ignore records with invalid/missing lat/lon/year.
- Group records into 4 collections:
  - Reptiles & Amphibians: Class in [Vertebrata, Reptilia, Amphibia]
  - Birds: Class == Aves
  - Mammals: Class in [Mammalia, Unplaced Mammals]
  - Fish: all other classes

2. Map
- Show an interactive world map with point localities.
- Color by collection:
  - Fish: blue
  - Reptiles & Amphibians: green
  - Birds: red
  - Mammals: yellow
- Points should be semi-transparent so overlap density is visible.
- Animation: points appear year-by-year from earliest to latest year.
- Controls:
  - Play / Pause
  - Speed buttons: Slow, Medium, Fast
  - Year navigation: First Year, Last Year, -1, -5, -10, +1, +5, +10
  - Loop toggle button (“Loop Time: On/Off”) that wraps playback to first year at end when on
  - Year slider (drag to any year, immediate update)

3. Header

- At top, show:
  - logo image CUMV_logo.png
  - title text: "Cornell University Museum of Vertebrates"
- Title font must be Times New Roman.

4. Line chart below map
- Plot cumulative specimen counts over time up to selected year.
- One line per collection with same colors as map.
- Add total cumulative line in black.
- Chart updates dynamically when year changes.

5. Design
- Entire app background must be pure white (#FFFFFF).
- Clean, professional, museum-quality UI.
- Responsive on desktop and mobile.

6. Run requirements
- Must run locally with:
  - npm install
  - npm start
- Serve on localhost (e.g., port 3000).

Implementation quality requirements:

- Use efficient rendering for many points (Leaflet canvas / layer strategy).
- Keep code organized and readable.
- Include concise comments only where needed.
- Include a clear README with exact Windows PowerShell run steps.

Results and Discussion

Two web applications were generated using AI-assisted coding based on datasets exported from Cornell University Museum of Vertebrates database: a search portal app for fish specimens (Fig.1), and an interactive map displaying sampling sites and accumulative specimen and lot numbers (Fig. 2). The applications performed the expected tasks, including filtering and exporting specimen records, visualizing collection localities on interactive maps, and displaying cumulative sampling data over time. Their development shows how AI-assisted coding can be an efficient way to lower the technical barrier to software development for natural history collections.

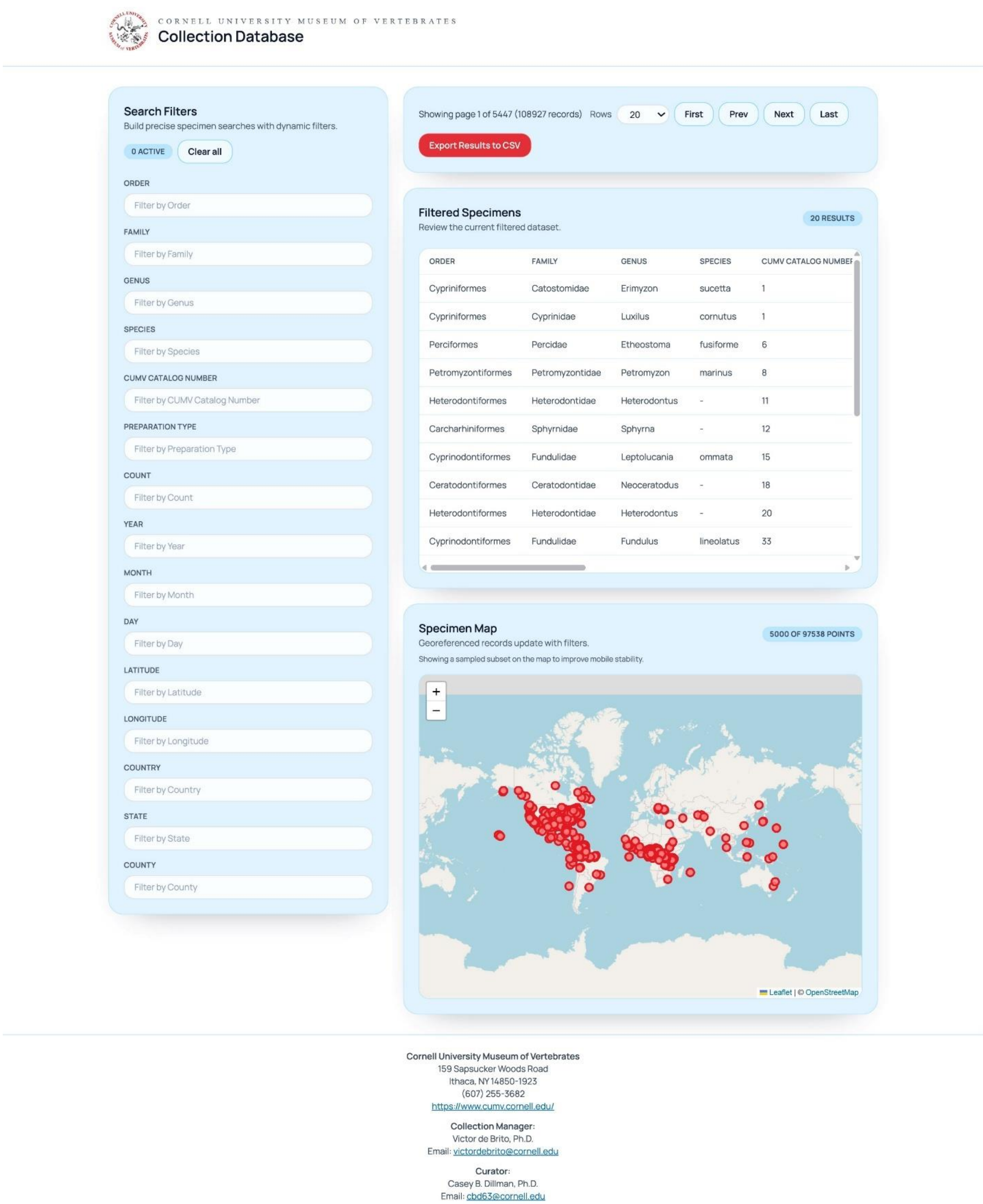

Figure 1. Collection database search portal developed for the Cornell University Museum of Vertebrates Ichthyology Collection. The interface allows users to filter specimen records, export

results as .csv file, and visualize selected records on an interactive map. The application is available at: https://debrito-victor.github.io/cornell-specimen-explorer/.

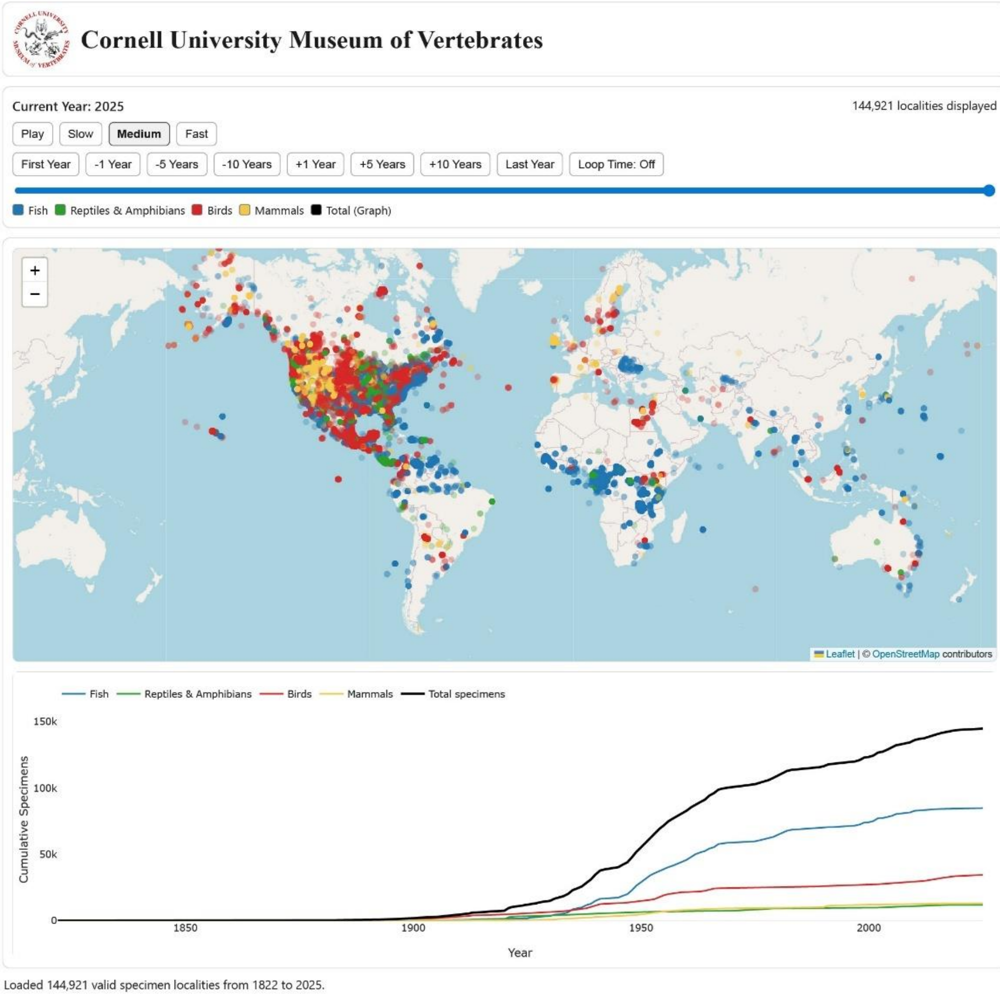

Figure 2. Interactive map displaying color-coded sampling localities for the four collections of the Cornell University Museum of Vertebrates, along with a cumulative line graph showing the growth of specimens and lots over time. The application is available at: https://debrito-victor.github.io/cumv-map/

Another benefit of this approach is the fast development. Once the data structure was exported from the collection database, working applications were generated and tested in less than 30 minutes. This rapid deployment allows collection staff to test different ways of presenting and exploring their data, leading to new tools for research, outreach, and education.

Despite these advantages, some caveats should be considered when adopting this AI-assisted coding. Large language models are stochastic by design, and they generate words (code in this case) by sampling from a probability distribution (Bender et al. 2021). The resulting applications varied in every execution, even in repeated uses of the same prompt. Clearly structured prompts that describe expected interface elements, functionality, data structure, and outputs in detail can greatly improve the quality of the generated code.

Posterior refinement is an important part of the AI-assisted coding workflow. Due to the lack of determinism, a functioning app with all desired features depends on the refinement process. In this multi-step process, the execution of subsequent prompts lead to improvements, debugging, and feature customization.

While AI-assisted coding lowers technical barriers, it remains most effective when users possess programming knowledge. AI can assist with app development, but the users are ultimately responsible for troubleshooting, maintenance, and deployment of the resulting tools. In this sense, museum staff adopting this approach should expect to be their own technical support.

Data security and system stability are also important considerations. Sensitive information should never be included in publicly accessible web applications, and tools should not be directly connected to live institutional databases. Instead, applications should be built using exported datasets or copies to reduce risks associated with security breaches or unintended data exposure.

Moreover, access to AI-assisted development platforms may require paid subscriptions, adding financial costs to museum application development. At the time of this project, free trial was available for ChatGPT Codex 5.3, but most AI coding tools will likely only be available through paid plans in the future. However, even if subscription costs become necessary, applications can be developed within a short period and maintained independently after the end of the AI subscription plan, minimizing long-term expenses.

Conclusion

The examples presented here illustrate how AI-assisted coding can be used to rapidly develop lightweight web applications tailored to the needs of Natural History Collections. Using an AI large language model, it was possible to create tools for data search, visualization, and outreach with relatively short development time and limited infrastructure. By bridging the gap between conceptual tool design and technical implementation, AI-assisted coding can enable a broader segment of the natural history community to develop customized digital resources. With careful implementation, this promising approach has the potential to accelerate the development of tools that support digitization, data management, visualization, and public accessibility.

Bibliography

Anon (2026) GBIF. Available from: https://www.gbif.org/.

Becker BA, Denny P, Finnie-Ansley J, Luxton-Reilly A, Prather J, Santos EA (2023) Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1. ACM, Toronto ON Canada, 500–506. https://doi.org/10.1145/3545945.3569759

Bender EM, Gebru T, McMillan-Major A, Shmitchell S (2021) On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In: Proceedings of the 2021 ACM

Conference on Fairness, Accountability, and Transparency. FAccT '21. Association for Computing Machinery, New York, NY, USA, 610–623. https://doi.org/10.1145/3442188.3445922

Boyer DM, Gunnell GF, Kaufman S, McGeary TM (2016) Morphosource: archiving and sharing 3-D digital specimen data. The Paleontological Society Papers 22: 157–181. https://doi.org/10.1017/scs.2017.13

Constable H, Guralnick R, Wieczorek J, Spencer C, Peterson AT, Committee TVS (2010) VertNet: A New Model for Biodiversity Data Sharing. PLOS Biology 8: e1000309. https://doi.org/10.1371/journal.pbio.1000309

Danks H (1991) Museum Collections: Fundamental Values and Modern Problems. Collection Forum 7: 95–111.

Davis CC (2024) Collections are truly priceless. Science 383: 1035–1035. https://doi.org/10.1126/science.ado9732

GitHub Staff (2025) Octoverse: A new developer joins GitHub every second as AI leads TypeScript to #1. The GitHub Blog. Available from: https://github.blog/news-insights/octoverse/octoverse-a-new-developer-joins-github-every-second-as-ai-leads-typescript-to-1/.

OpenAI (2026) GPT-5.3-Codex. Available from: https://chatgpt.com/codex.

Rocha LA, Aleixo A, Allen G, Almeda F, Baldwin CC, Barclay MVL, Bates JM, Bauer AM, Benzoni F, Berns CM, Berumen ML, Blackburn DC, Blum S, Bolaños F, Bowie RCK, Britz R, Brown RM, Cadena CD, Carpenter K, Ceríaco LM, Chakrabarty P, Chaves G, Choat JH, Clements KD, Collette BB, Collins A, Coyne J, Cracraft J, Daniel T, De Carvalho MR, De Queiroz K, Di Dario F, Drewes R, Dumbacher JP, Engilis A, Erdmann MV, Eschmeyer W, Feldman CR, Fisher BL, Fjeldså J, Fritsch PW, Fuchs J, Getahun A, Gill A, Gomon M, Gosliner T, Graves GR, Griswold CE, Guralnick R, Hartel K, Helgen KM, Ho H, Iskandar DT, Iwamoto T, Jaafar Z, James HF, Johnson D, Kavanaugh D, Knowlton N, Lacey E, Larson HK, Last P, Leis JM, Lessios H, Liebherr J, Lowman M, Mahler DL, Mamonekene V, Matsuura K, Mayer GC, Mays H, McCosker J, McDiarmid RW, McGuire J, Miller MJ, Mooi R, Mooi RD, Moritz C, Myers P, Nachman MW, Nussbaum RA, Foighil DÓ, Parenti LR, Parham JF, Paul E, Paulay G, Pérez-Emán J, Pérez-Matus A, Poe S, Pogonoski J, Rabosky DL, Randall JE, Reimer JD, Robertson DR, Rödel M-O, Rodrigues MT, Roopnarine P, Rüber L, Ryan MJ, Sheldon F, Shinohara G, Short A, Simison WB, Smith-Vaniz WF, Springer VG, Stiassny M, Tello JG, Thompson CW, Trnski T, Tucker P, Valqui T, Vecchione M, Verheyen E, Wainwright PC, Wheeler TA, White WT, Will K, Williams JT, Williams G, Wilson EO, Winker K, Winterbottom R, Witt CC (2014) Specimen collection: An essential tool. Science 344: 814–815. https://doi.org/10.1126/science.344.6186.814

Rohwer VG, Rohwer Y, Dillman CB (2022) Declining growth of natural history collections fails future generations. PLOS Biology 20: e3001613. https://doi.org/10.1371/JOURNAL.PBIO.3001613

Specify Collections Consortium (2022) Specify 7. Available from: https://www.specifysoftware.org/.

Suarez AV, Tsutsui ND (2004) The Value of Museum Collections for Research and Society. BioScience 54: 66–74. https://doi.org/10.1641/0006-3568(2004)054%255B0066:TVOMCF%255D2.0.CO;2