CHAPTER 15

User Interfaces for Creating Digital Research

Tarrin Wills

People working in the humanities often do repetitive and structured analyses of texts, images, or other cultural products. Often these processes could be done through digital methodologies, but more often than not, humanities scholars and students choose to do them in a non-digital way, limiting the way their work might be used by others. These choices are difficult to understand for those of us working in Digital Humanities.

The problem can be addressed in part by paying attention to the way **applications** and interfaces are used by humanities scholars to interact with information in their particular fields. Projects can develop methods and interfaces that allow students and researchers to feel comfortable when producing and analyzing data, while maximizing the potential of that data for research purposes. "Skaldic Poetry of the Scandinavian Middle Ages" is a Digital Humanities project that exemplifies small-scale custom interface solutions. This chapter focuses on the technologies and applications that connect scholars who produce data with the data itself, namely, the **user interface** (UI).

T. Wills (✉)
University of Copenhagen, Copenhagen, Denmark
e-mail: tarrin@hum.ku.dk

l. levenberg et al. (eds.), *Research Methods for the Digital Humanities*,
https://doi.org/10.1007/978-3-319-96713-4_15

A Short History of UIs and Digital Humanities

In the late 1980s and 1990s graphical interfaces emerged for creating digital documents. In particular, **word processors**, such as Microsoft Word, which displayed a close approximation of the print document being created on screen developed 'what you see is what you get' (WYSIWYG) interfaces. This development rapidly promoted the use of computers in all fields where documents were used and exchanged. Humanities research was no exception, and the technology, which is now ubiquitous, highlights one of the important factors in a successful user interface: it is much easier for a user to create data if they get very accurate and quick feedback on how the data will be displayed and understood. In other words, the WYSIWYG interface not only allows the user to accurately create formatted textual data, but also brings that data as close as possible to what the end-users (their audience) will need to see: the printed page.

Word processors have limitations for working with research data. A researcher could use a word processor to produce a scholarly edition, a dictionary, or a catalogue suitable for print publication, but they would lose much of the information about the structure and meaning of the resource. The kinds of useful information which cannot be retrieved easily from a word processor file might include: a particular version of a text in an edition, dictionary entries organized by part of speech instead of alphabetically, or the cross-references to texts in a catalogue of manuscripts. These situations mean that electronic documents created by word processors are often little better than print publications in terms of exchanging data.

The solution to these problems was to develop standards, such as the Text Encoding Initiative (**TEI**) standards, that would record the semantic information and structure.[1] Data produced using these standards could form the basis of print and electronic publication by using programming languages or style sheets to transform the code into more readable documents or web pages. To do so, the data creator must have a close understanding of a complex technical standard and the ways it can be transformed. TEI and similar standards have been widely adopted and

[1]The Text Encoding Initiative (TEI) released its first official standard in 1994, http://www.tei-c.org/About/history.xml.

most users of TEI directly create the (normally) **XML** code themselves, although often with the aid of applications that check the validity of the code. Using these methods the data creator has very close control over the semantics (the meaning of the data), but the trade-off is that the code must be separately processed to make it useful to someone who does not understand it. The main question that this chapter addresses is how to manage and minimize the trade-offs between the integrity and usability of the research data, the needs and methods of the data creator(s), and the needs of the end-users and research field more generally.

The Data and the User

This chapter has already mentioned two types of interface for data creation, namely WYSIWYG word processors and editors for producing XML code. These represent extreme ends of the spectrum between data accessibility and integrity. There is (almost) no scholar alive today who is not comfortable with creating a word processor document, so in many ways this technology represents maximum accessibility. **TEI/XML** documents represent the pinnacle of data integrity, combining a well-documented de facto standard with detailed semantic and metadata tagging, but very few scholars end up directly creating research data using such a standard.

Depending on the detail of the information required, the two types of encoding may be exchangeable within a single project. Consider the following: a convention for typesetting transcriptions of manuscripts is to expand abbreviations using italics, e.g.: 'gefn*ir*' (expanding the abbreviation for er/ir in 'gefñ' in a manuscript).[2] TEI might encode this using the expansion tag: 'gefn<ex>ir</ex>'. Someone with knowledge of manuscript transcription conventions but not digital text encoding would be able to correctly interpret the italics, but not the code. Conversely, someone with an XML background but no transcription experience may not understand the italics, but could interpret the code by looking up the TEI standard. The TEI/XML text would ideally be used for archival and exchange purposes. But, a digital transcriber need not necessarily understand this format: if the italics unambiguously represent expansions of

[2] Manuscript 1009 fol. in the Old Royal Collection (GKS), Royal Library in Copenhagen, fol. 2v/22.

abbreviations, they can be transformed at the point of storage into TEI/XML code and can be retrieved through the interface as italics. Even easier to transform in this way are different types of brackets, which might, within a particular project, indicate expansions, editorial interventions, text supplied from other sources, damaged or difficult sections, and so on.

TEI allows the encoder to include both the abbreviated and expanded form in the above example. If that level of detail is required, where parallel information is encoded at the same point, it becomes much harder to convert between a word processor format and XML. Nonetheless, a user interface might be able to find other ways, such as through a pop-up dialogue, to enter the additional information. The information might then be hidden under normal circumstances, or in print, but available when the user clicks on a word in a web page, for example.

A project might use different methods and standards to give a variety of end-users outputs in appropriate formats. For a digital repository such as the Oxford Text Archive or Menota, the data should be in a standardized, exchangeable format such as TEI. This allows other digital scholars to use the data with reference to a common standard, but without requiring specific disciplinary knowledge to understand the data and its structure. A researcher working in the discipline (e.g. a historian or linguist) may want the data in a traditional format such as a book, or a format which they can print or view on an e-book. Such a format can also be stored in a library without the pitfalls of digital-only archiving. Other researchers may be able to make use of an interface that allows them to query the data, transform it, or combine it in ways that they can use for research purposes. Some projects might have the general public in mind and create an interface which allows the public to access the information in an accessible form. All these types of output can be created from a single data repository, using different techniques to transform the data as necessary.

The skills and interests of the end user are highly diverse, but we can reduce them to four types:

1. Digital specialists without a background in the specific discipline of the project;
2. Discipline specialists with a reasonable comfort level with digital approaches and interfaces;
3. Discipline specialists with a low interest in digital interfaces;

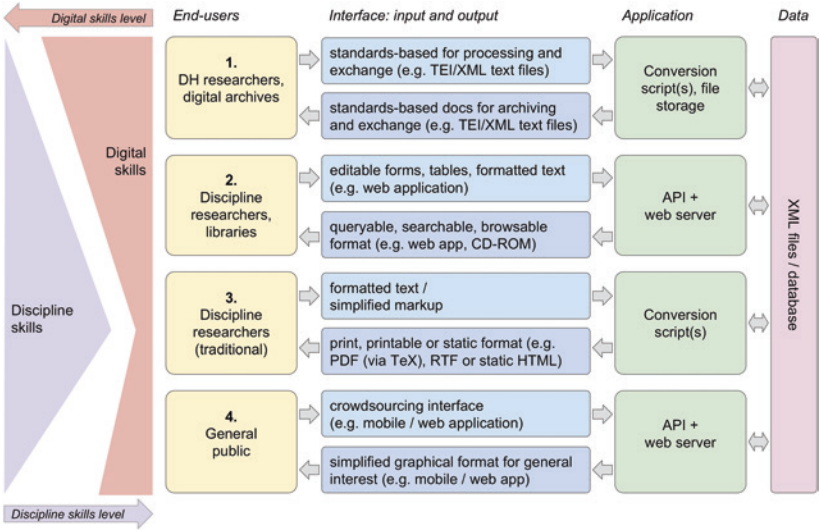

Fig. 15.1 Systems of conversion and skills required for different end-users

4. The general public, who may be interested in accessing the information digitally but who have little or no training in the particular subject.

For each type of user, the interface to the data might be organized in a different manner with an application converting the data from the form used by the project to the form required by the user. This model is shown in Fig. 15.1. (It should be noted that there is no restriction on how input formats might be exported in various output formats.)

The left of the figure also demonstrates the trade-off between disciplinary skills and digital skills. There is no inherent trade-off, but in practice this tends to be the case. A traditional print outlet in a particular discipline, such as a historical dictionary or a scholarly edition of a book, relies on the conventions of print media, which can be explained further in an electronic interface, but are already familiar to someone with greater disciplinary skills, who is used to those conventions.

The figure also represents different ways of inputting information for a project. Different end-users, if they are supplying the data, will need different methods for producing the data. A student might submit a

TEI/XML document that is then processed by the project's application (as in the Menota text archive at clarino.uib.no/menota); a researcher with more limited digital interests might retrieve and edit the data using web forms or another type of interface; formatted text might be submitted by someone really uncomfortable with anything beyond a Word file, but if it is consistently formatted then it could be transformed into a standard technical format; and, a member of the public might be able to submit data using a more restrictive, but user-friendly, platform. The same project may use some or all these interfaces, depending on the audience and the contributors. The case study outlined below (The Skaldic Project) will illustrate how many of these approaches work.

Project Requirements for Selecting Interfaces

There are a number of practical issues which need to be considered before an interface is chosen. The number and variety of users will have an enormous impact on the solution required, regardless of the project. The first question is therefore about the diversity of users:

1. *Who will be working on the data? Will more than one person be working on similar data at overlapping times?*

If you are the only person working on the data for the duration of the project the solution can be very simple, as it may be a matter of learning the necessary technology. As soon as there is more than one person, there is the problem that users may have different approaches, skills, or interests in technology, and may need to edit the data at the same time.

2. *Where will the users be? At the same workplace? Working internationally? In the field? Do they have access to the same technology (software and hardware)?*

People working at the same workplace will normally have access to shared storage, so they can access the same data simultaneously. This makes it much easier to determine what solution is used, as it is a matter of choosing an application for a particular operating system and storing the data on a network drive. Security is largely taken care of by the institutional infrastructure. Outside of a single organization there are a number of other issues to deal with, including security, and software and

hardware support. If data will be created or edited in a fieldwork environment, researchers need to consider whether the users will have network access, the kinds of devices they will be able to use, and how much information they need to support data creation.

3. *What is the nature of the data to be created? Is it, for example, textual, geographical, or visual? Do different types of data or media need to be linked together? Are there external resources that should be linked, such as online gazetteers and registries?*

These issues are dealt with elsewhere in more detail in this book, but some types of data require more graphical or interactive interfaces than others. Information that can be found digitally, in an authoritative form, should link to the source in a way that facilitates updates. The interface needs to be able to handle all the necessary types and sources of data.

Solutions to the problems outlined here (including, multiple users, semantic tagging of information, multiple uses and contexts of the data, simultaneous updating, and so on) have been solved most spectacularly by social media. It is in the nature of a social network to try to bring together those creating information and their audience. Social networking platforms such as Facebook and Twitter incorporate a range of external sources of information provided implicitly or explicitly by the user: geographical information (location, from device or photos), links to external resources (web links shared), semantic tagging (hashtags), networks (friends/followers) and links to networks (tagging friends/handles), and metadata about the users themselves (profiles). These are applied to text, images, videos, links, and other types of resources. This information is used largely to generate targeted advertising, but it is worth considering how such interfaces facilitate the supply, linking, and categorizing of a very broad range of media and data.

Interface Solutions

There are a variety of interface solutions, which range from simpler to more complex depending on the type of users for the project and the kind of data to be created. The following is a list of user interface solutions depending on the complexity of the project, the number of contributors, the type and complexity of the data, and the storage format for the data.

Single User, Desktop Environment, Data as Files on Computer

If your project has one user working on one type of data at a time, there are several desktop tools with easy to learn interfaces, many of which are free and open source. These solutions do not rely on creating new interfaces or systems. Many of these applications are mentioned elsewhere in this volume: e.g. text encoding (Oxygen), statistics (R, Gephi), geographical data (ArcGIS, QGIS), tabular data (spreadsheet applications), relational data[3] (Access [Windows only], FileMaker). Note that word processors are not considered appropriate solutions in Digital Humanities in themselves, because they do not structure data in a way that can be further processed.

Multiple Users, Desktop Environment, Data as Files on Network

A certain amount of complexity occurs when you have different people working concurrently on the same data or on data that have to work together in some way, such as when it is processed by the same application. If you are all working together in the same location or can otherwise communicate directly, the solution can be as simple as a manual division of labor and manual data management. This might work by using the applications described in the previous section, but storing the files generated in a collaborative repository such as Dropbox or Google Drive. This type of solution is widely used (e.g. projects such as the Menota Handbook [menota.org], Stories for All Time [fasnl.ku.dk]) and requires manual management of file storage, so that the data are retrievable and processable by the project's applications and by other members of the project. Participants in the project should define files naming protocols and the folder arrangement in the drive. This means that an XSLT application, for example, can generate web pages from the contributors' XSML files. Individual files, however, cannot be edited concurrently without the risk of data being overwritten. So, normally, such a project would divide the files and work between different members of the project.

Code repositories such as Sourceforge, Github and Google Code provide interfaces, which allow contributors to safely check out, edit, and check in files as well as other features for managing collaborative projects. This means that files are not overwritten by mistake and changes

[3] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM* 13, no. 6 (1970): 377–387.

can be tracked, particularly if text files are used by the project, such as XML and programming code. Compiling and processing of the files, however, needs to be done off the site, which may require some projects to develop scripts or applications for downloading the files in addition to processing them. A number of Digital Humanities projects use these repositories for managing code, particularly where the code is intended for programming libraries or applications.

Multiple Users, Desktop Environment, Data on Network

When you have multiple users working on the same data or data structures at the same time the normal solution is to use a networked database. Structured Query Language (SQL) is the standard that governs how data is retrieved, stored, and managed in such databases. There are various free and open source SQL databases, which can be used to implement this solution. Some desktop or web applications (such as GATE Teamware [gate.ac.uk/teamware]) will set up a database for you, meaning that there is little extra work except in learning how the application connects to the database, so that other users can access the data. Geographical (GIS) applications (such as ArcGIS and QGIS) work with data in tables that can be stored and shared on various SQL database server applications.

Multiple Users, Web Environment, Data on Network

Where the data used by the project is relatively simple, such as data in tables or geographical data, there may be cloud solutions to edit and store the data using web or mobile applications. Google provides a few free services which allow multiple users to edit tabular data (Sheets, Fusion Tables) or geographical data (My Maps, Fusion Tables) on different devices. There are often web alternatives to desktop research applications, such as RStudio for the R statistics application and GATE Teamware for the GATE language processing application. All these solutions allow for different people to work simultaneously on the same data in different locations and on different devices.

Other applications may require the data to be stored in a networked (SQL) database. There is a powerful web open source application for MySQL and MariaDB called phpMyAdmin, which can be used to set up these databases (the equivalent for PostgreSQL is phpPgAdmin) and

enter and edit information online for different users. Most universities provide a MySQL or similar server for staff and students, which can be used for these purposes, although it is increasingly common to use virtual servers for SQL databases.

Multiple Users, Custom Desktop Application, Data on Network

In situations where you are dealing with very complex data, the available software applications may not be able to link together and structure the data adequately for the research purposes of the project. If people are working at the same workplace, where equipment, software, and support are consistent, a desktop application may be the most appropriate interface for entering data. Such systems are often used by larger museums to manage information about their collections, or long-term projects such as dictionaries.

The screenshot in Fig. 15.2 shows an interface to the Dictionary of Old Norse Prose's database.[4] Visual information (here the original citation slip and a scan of the printed edition, which the citation is taken from) is provided to assist the lexicographer in organizing citations within an entry. The interface is a desktop application that connects to an Oracle SQL database server.

Desktop applications tend to be highly responsive to user input, as there is little time lag in connecting to the database and very fast transfer speeds between the application and the data source. Additionally, the code is normally executed natively by the operating system, rather than by an intermediate application such as a browser. This often makes the interface much faster to respond to user input than a web application, but the user needs to have access to the required network and operating system, and to be able to install the software for any device they need to use.

Multiple Users, Custom Web Application

In situations where there are different users in different locations using varying devices, a custom web application is often the most powerful and flexible solution. As mentioned previously, social media sites are examples

[4] James Knirk, Helle Degnbol, Bent Chr. Jacobsen, Eva Rode, Christopher Sanders, and Þorbjörg Helgadóttir, eds., *A Dictionary of Old Norse Prose/ Ordbog over det norrøne prosasprog* (Copenhagen: The Arnamagnæan Kommission, 1989).

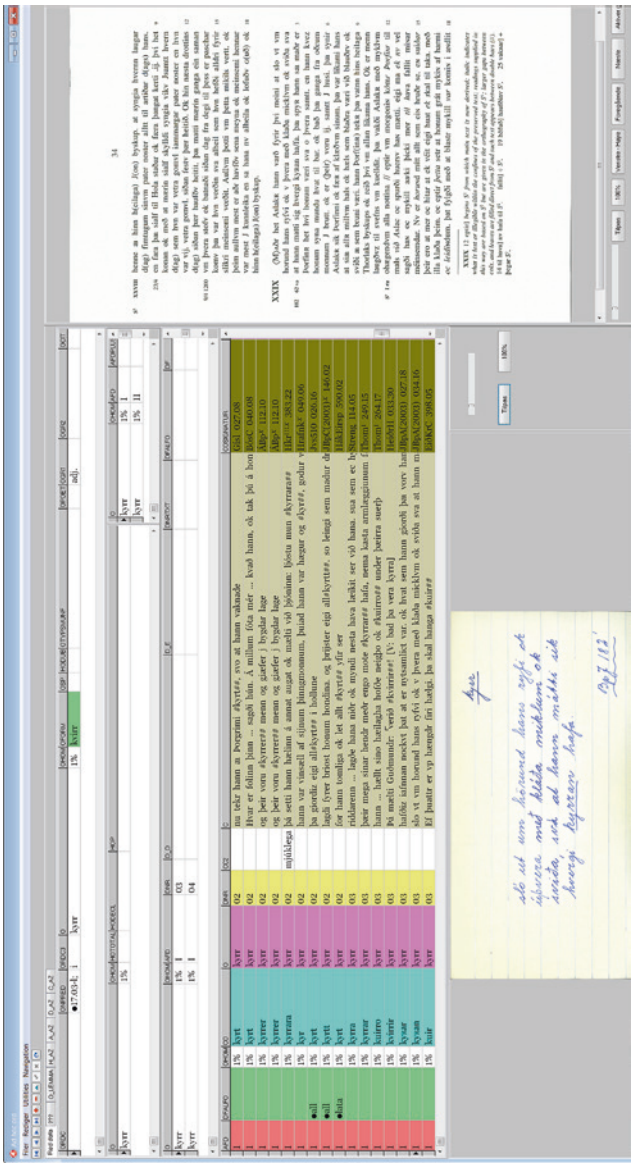

Fig. 15.2 Dictionary of Old Norse Prose: desktop application for constructing entries

of this type of solution, where multiple users are able to access, contribute, and link information. Social media sites represent the enormous **scalability** of the technology, to the extent that hundreds of millions of users can simultaneously edit and link within a single database.

The popularity of this model means that the technologies it uses (database, programming interface and website framework) are free and open source, and are accompanied by a massive amount of support available online. This solution, the custom web application, is discussed at length in the next section.

Crowdsourced, Web Form (No Editing)

Situations where data is collected from non-specialists is often called **crowdsourcing**, but traditionally researchers have often gathered data from the public for research through surveys and forms. Simple digital forms—where the input is in the form of text, yes/no answers, and multiple-choice options—can be created using a variety of free online tools. The information is normally stored in a tabular format such a spreadsheet, which can be used for statistical and qualitative analysis. This type of interface involves single direction information gathering from the end-user's perspective: they submit the information and, once submitted, it cannot normally be edited it. Because the interface is one-way, it is much simpler to create than a system that allows the users to retrieve and edit the information themselves.[5]

Crowdsourced, Custom Web Application

A custom web application may be appropriate in situations where a project is **crowdsourced**, but either requires more complex input than traditional forms (e.g. where images are involved, or linking of data) or a more complex process, where contributors can retrieve and edit their contributions. These interfaces are the most difficult interfaces to develop because they need to be understandable and usable to a very large variety of people with little or no training in the particular discipline. The interface needs to be very clear and simple, with workflows that can be followed by members of the public. Any inconsistencies or faults may put people off from contributing, thus negating the benefits

[5] Note that you may require ethics approval before conducting any research that involves gathering any kind of personal information from the public.

of crowdsourcing. The application must be sufficiently flexible to allow complex input, but at the same time there must be systems in place to deal with deliberate misinformation supplied by contributors.

Such projects are normally limited to particular processes that are more reliably or easily done by humans than machines, such as transcribing handwritten documents. Projects such as Zooniverse (zooniverse.org) provide support for developing crowdsourced research projects. At the time of writing all the humanities projects on Zooniverse involve transcribing and annotating manuscripts and other documents, a process that is still easier for humans to do than computers. For example, one project asks users to identify images in historical scientific publications and annotate them according to captions and keywords. Another example of a largely crowdsourced project is the Textual Communities project, which allows signed-up users to contribute XML transcriptions of manuscripts.

This latter type of application gives users much more control over their contributions and credits contributors. It is limited to producing XML code directly (with syntax highlighting to assist) and does not link to detail outside of the page. There are several advantages to giving users this amount of control and credit: they have an incentive to provide more and better-quality input if their responsibility is acknowledged. Additionally, contributions can be included as assessed work in taught courses, or as part of other research projects. There is added complexity in such an interface, because user access and privileges must be carefully managed.

Web Interfaces for Creating and Editing Data

The following is meant as background to creating a web application as a user interface for a Digital Humanities project. A web interface sits at one end of a complete web application providing the necessary information and context to assist input, dealing with the user input, and feeding back information about the input. A user opens a web site by inputting a URL in a browser. The web server executes an application, which produces a web page. The resulting web pages normally have links and/or forms that allow the user to continue the process by opening new pages. The information that is sent may be used to store new information on the server, to retrieve information, or a combination of both.

Web applications can be very complex because they require a knowledge of the structure of the underlying data (e.g. XML, relational data), the ways in which the data can be accessed and transformed (e.g. XSLT, SQL), the programming language used to make the transformation (the

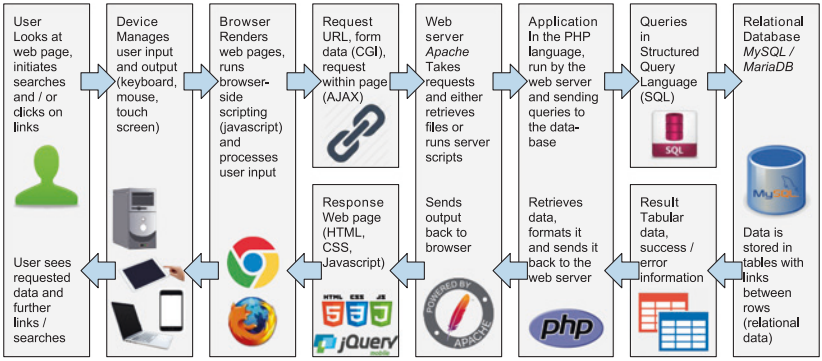

Fig. 15.3 Stages of a web database application and languages used

Application Programming Interface (**API**) language), as well as the web output technologies (normally HTML, CSS and Javascript) used to generate the resulting pages. Fortunately, many of the issues are solved by a range of libraries and plugins, which various projects and initiatives have made public and continue to support.

Figure 15.3 shows the structure of the application from user to database and the various technical languages commonly used at each stage of the process, although the application will tend to focus development on one of the stages (web, API or SQL). There are many libraries and frameworks, which simplify this process. Figure 15.3 represents a configuration widely used by large applications wherein users both contribute and access data. It is the model (with minor variations) used by Facebook, Twitter and Wikipedia, for example, which all specifically use MySQL/MariaDB as the database server and PHP as the API language. Alternatively, an XML-based project might use XLST to read and transform XML data, for example, and there are many other languages which can be used to write the application itself.

Example of a Complex Application: The Skaldic Project

The project Skaldic Poetry of the Scandinavian Middle Ages (or just the Skaldic Project) aims to edit, translate, and annotate a major corpus of poetry from early Northern Europe. It illustrates many of the small-scale solutions that might be implemented in developing a custom interface. It represents some of the limits of what a single researcher-developer can

produce over a number of years, or what a small team could create in a shorter period.

There are several complexities to the project's corpus: it is preserved in several hundred manuscripts and is attributed to hundreds of different poets; there is considerable variation between the manuscripts, as well as variation in how previous editors have reconstructed the poems; and, the poetry is largely preserved piecemeal, in prose works, which have their own transmission history. And, not least, the poetry itself is highly complex, requiring a reordering of the syntax to bridge the gap between the verse and the translation, as well as glosses of very complex and variable poetic expressions (known as *kenningar* and *heiti*). The project is led by 5–6 senior scholars, including the author of this chapter, who oversee contributions from over 40 different editors and are assisted by over a dozen researchers funded by various sources. The project leaders and assistants are based in the US, Europe, and Australia, meaning that there is 24-hour use of the project's digital resources. The end-point of the project is both a traditional print publication in the form of a book series as well as an interactive digital edition. The data structure of the resource therefore had to be designed to encompass sufficient information in order to structure and format the edition in the desired forms.

The web interface originally began as a tool to assist contributors in finding contextual information about their texts. As web technologies improved, it became quickly apparent that a web interface could be used for the processes of entering and exporting the edition, as well as searching, browsing, linking, and analyzing it. In order to deal with the level of linking required by the project, the original TEI/XML structure of the text was translated into a relational data model.[6] The resulting structure stores the text as words and lines in tables, rather than as marked-up text. This means that the interface needs to be able to edit the information in tables and output the text in an easily readable form to the user. Web development **frameworks** can effectively create applications that can be used on all devices, such as Bootstrap and (with some modification) JQuery Mobile 1.4.

Figure 15.4 shows the same page adapting for the screen width, using features of JQuery Mobile and some custom CSS and Javascript. The first shows how the page appears in a desktop browser. On a narrower device such as a tablet the left menu is removed and the header

[6]Tarrin Wills, "Relational Data Modelling of Textual Corpora: The Skaldic Project and Its Extensions," *Digital Scholarship in the Humanities* 30, no. 2 (2015): 294–313.

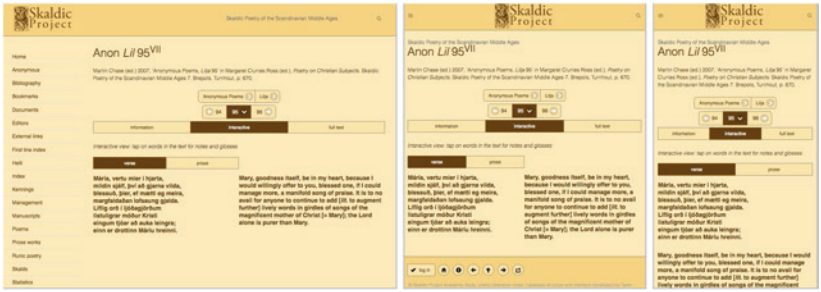

Fig. 15.4 Adaptive web design interface

rearranged, so that the menu is accessed by a button at the top left. On a very narrow device such as a phone the text boxes are **reflowed**, so that they appear above and below each other. Additional information is accessed through pop-ups (e.g. information about each word), and links and buttons are appropriately sized for both mouse and touch interaction.

The advantage of an Adaptive Web Design approach is that the same application can function as a web application for desktops, and a mobile app for tablets and phones. Supplying icons for the two main mobile platforms (iOS and Android) allows users to easily add a link to their phone app list and effectively makes the site function as a mobile app.

The Workflow

Many of the leading researchers in the Skaldic Project were unfamiliar with digital technologies and had previously only worked on projects based on producing word processor files for print publication. Rather than requiring them to learn entirely new processes and systems, the project defined a way in which the data could be submitted as word processor documents using simple but consistent formatting and markup, so that they could be transformed by assistants into the required format for the digital resource. The format for submissions, therefore, had to be structured so that the data could be entered into the digital resource, containing all the information required with some additional formatting and markup. Editors could also choose to use the web interface to enter their editions. The final printed volume is similar to the files provided by most editors, except that the submitted files contain a few additional

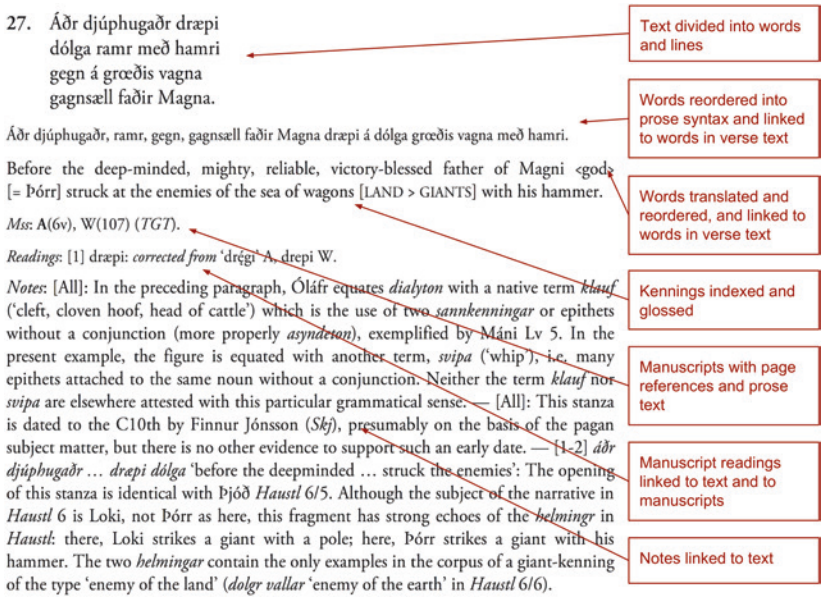

Fig. 15.5 Example of printed version of the Skaldic Project's editions

simple markup techniques designed to help the application process some of the structured information.

Figure 15.5 shows both a simplified version of the start-point and the exported printed end-point of the process. In the process, a rich digital resource is created as the implied structures of the edition are put into a series of linked tables. There are a series of forms which assist in this process, as described below.

Step 1: Inserting the Text

In the first stage, a form takes plain text in lines and adds it to the database structure as rows in the tables of words and lines, linking each together and numbering them for later ordering and referencing. It processes simple markup representing emendations (where the editor has changed the text from the manuscripts) using angle brackets and asterisks corresponding to the TEI <corr> element.

This method illustrates one way of entering data, where the application takes a simple input and creates a complex data structure. This process is not easily reversible: if the user was able to edit the text the way it was

entered, it would risk information loss. For example, if a word's spelling was changed or the word boundaries altered, the data linked to a given word (such as the translation, variants, or notes) may become incorrect. Some other projects use a similar process to allow contributors to create transcriptions with simple markup and transform it into TEI/XML.

Steps 2 and 3: Reordering to Prose Syntax and Linking the Translation
The previous form creates a series of rows in the tables containing lines and words in the database. Once created, the tabular data can be edited using a generic form for editing rows and columns in a table (this form, as with the others in this section, changes format according to the device used). The application includes generic methods for building forms from the tables in the database, giving appropriate inputs for the types of data in each column (e.g. text, numbers, links). The form shown here takes all words linked to a particular stanza and allows the user to adjust the columns which store the information about the prose syntax and order. A Javascript plugin allows drag and drop reordering of the columns. Figure 15.6 shows a word in the process of being moved. Another feature shows the resulting prose word order in the box above the table, giving instant feedback to the user about how the text will appear when outputted. This is very important given that the row/column structure of the data looks very different from the text output, which is in sentences.

The translation is entered using a similar form to the prose word order, allowing the user to reorder the words to form an idiomatic English translation. The words are linked to the original text, so that any

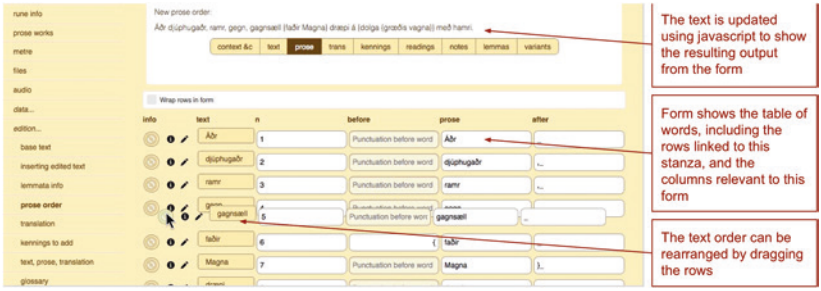

Fig. 15.6 Form for rearranging text into prose syntax

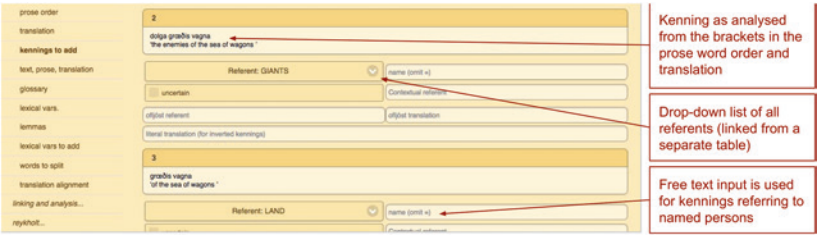

Fig. 15.7 Form for entering kenning analysis

word in the corpus will have a contextual gloss. And, another simple markup convention is used here to indicate the structure of **kennings**, namely, curly brackets. These are processed at a different stage and do not appear in the final edition, but the brackets provide sufficient information to the application to process them (Fig. 15.7).

Step 4: Analyzing the Kennings
Here the application analyzes the inputted brackets to establish which words in the prose order and translation contain kennings. The user is then prompted with a form for each kenning, allowing them to gloss the kenning and provide additional information if necessary. Here the kenning 'sea of wagons' refers to 'land' and 'the enemies of the sea of wagons' (i.e. the enemies of land) are 'giants'. This information is linked, so that an index of kennings is automatically available and an end-user can find these among all kennings for 'land' and 'giant.'

When the form is first used the kennings are generated from the markup in the text and, when updated, the application adds them to a separate table with links to the main text. Subsequent views and updates of the form take the information from the index table rather than from the markup in the inputted text, but they are presented with the same interface (Fig. 15.7).

Step 5: Variant Readings
The variant readings are entered as a separate table where each row links together one or more words in the text to one or more manuscripts, which are witnesses to the text. The manuscripts are entered separately and the user can select each word and each manuscript, which adds the variant reading into a text box. This form uses a generic interface, which

Fig. 15.8 Form for entering manuscript variants

defines how each column in the table is to be edited. The database uses arbitrary numbers to link rows in different tables, so the form defines how the linked data (words and manuscripts) are to be represented in the form based on the information in the linked tables. This information is used to show a traditional variant apparatus in the printed book, as well as an interactive apparatus in the web interface (Fig. 15.8).

Step 6: Annotating the Edition

The notes form links words in the text to annotations. The discursive content of the notes (i.e. sentences) are formatted using a restricted set of HTML tags. This enables simple formatting, but also the identification of bibliographic and internal references, which are later processed so that links and pop-ups can clarify the abbreviated references.

The text is entered using a WYSIWYG Javascript plugin called TinyMCE, which has been set up to restrict the formatting options. The entered text is processed when it is sent to the server to convert the HTML formatting into semantic tags, and to generate links from bibliographic and other references. The same plugin is used to enter other formatted text into the database including introductions to poems and volumes, biographies, and contextual notes.

The WYSIWYG html solution gives quick feedback to the user and leaves the semantic processing to the application. It can be used in a project such as this, because the project uses well-defined conventions about the meaning of formatting styles such as italics (e.g. for references to texts) and superscripts (for references to other poems in the corpus).

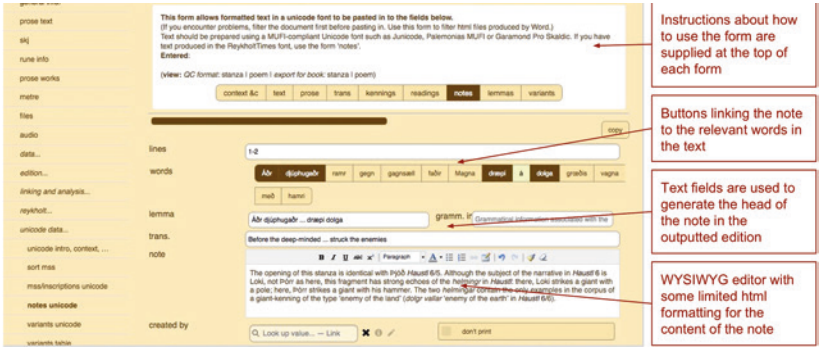

Fig. 15.9 Form for entering notes to the text

While the input (and output) uses simple formatting, simple inputs can be used to create unambiguous and meaningful data for an individual project with clear internal conventions (Fig. 15.9).

Step 7: Linking to Dictionary Headwords and Other Processes
While not part of the printed edition, the process of linking each word to dictionary headwords (lemmatising) allows for lexical and linguistic analysis of the corpus. In a highly inflected language such as Old Norse, there are normally a very large number of grammatical forms and spellings for any dictionary headword. This form (Fig. 15.10) aids the process by remembering previous choices and supplying information about both the word in the text and the dictionary headwords. The form presents each word as a row in a table, attempting to find the possible headwords for each word based on previous input and showing them as a list. Users can click/press various buttons to show pop-ups with information about the word or lemma (in the screenshot above, other words linked to the headword are shown). If the user needs to enter a lemma which is not automatically shown, they can type part of the word into the search box (also shown in the screenshot). These features—the pop-ups and search boxes—are implemented using JavaScript, where smaller pieces of information are fetched from the server. There are other forms (not shown) designed to assist in other processes of altering the table structure of the text, including splitting words (creating two rows out of one in the words table), joining words, and adjusting the alignment of the text and translation.

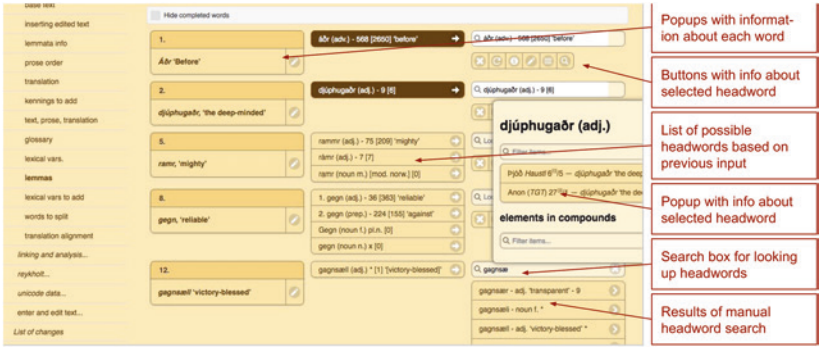

Fig. 15.10 Form for linking words to dictionary headwords (lemmatizing)

Designing the Interface

The design of user interfaces is the subject of an international standard (ISO 9241), which covers all human-machine interfaces. The standard describes, in great detail, standard ways in which information in the interface should be organized and formatted. These principles of presentation are fairly self-evident and include: clarity, discriminability, conciseness, consistency, detectability, legibility, and comprehensibility. Normally a project will use a pre-existing framework or template and adapt it to its own purposes. Pre-existing frameworks such as Bootstrap and Omeka provide structures for arranging information in a web interface according to these principles, but attention should still be paid to the clarity and conciseness of the specific application's content.

The user interface always involves a **dialogue** between the application and the user, where the application asks the user for input and gives feedback in response to input and information to assist in the input. There are seven principles that outline how the process of interaction should work in relation to the user's knowledge and expectations on one hand, and the required result of the process for research purposes, on the other. These principles are outlined below, with some examples from the Skaldic Project case study.

1. Suitability for the task: the interface should effectively work towards completing the desired outcome of the process. In our case study, the end result is the digital edition with accurate linking of text, translation, and other features.

2. Self-descriptiveness: the input required and the output desired by the user should be understandable at the point it is used, providing immediate feedback automatically (such as through real time updates based on the form input) or as requested by the user (such as through pop-ups).
3. Controllability: the interface should allow the user to determine the speed of the process, as well as go to earlier or later stages in the process. In our case study, this is achieved by separating the stages of entering the edition into separate forms, which can be revisited to modify previously submitted information.
4. Conformity with user expectations: the interface should be consistent; be sensitive to the knowledge, expectations and digital proficiency of the user; and, use conventional terminology and formatting wherever possible.
5. Error tolerance: the interface should prevent users from entering input erroneously, but also where possible it should be able to handle errors from the user, minimizing additional input. Various filters in the Skaldic interface, for example, take the free text input and remove extraneous formatting to ensure the stored data is semantically unambiguous and simple.
6. Suitability for individualization: the interface should be modifiable according to the user's needs, preferences, and knowledge of the system or discipline. The Skaldic interface makes common processes easily available, but allows additional and more complex data to be edited through additional forms. A bookmarking feature also allows users to easily return to pages and forms.
7. Suitability for learning: the interface should guide the user to help them learn the processes and thus improve speed and accuracy of input.

These principles are underpinned by a body of research into user interfaces.[7] By far the easiest way to begin designing a user interface,

[7] Mitchell Whitelaw, "Generous Interfaces for Digital Cultural Collections," *Digital Humanities Quarterly* 9, no. 1 (2015), accessed May 5, 2018, http://www.digitalhumanities.org/dhq/vol/9/1/000205/000205.html; Iwe Muiser, Mariet Theune, Ruud de Jong, Nigel Smink, Rudolf Berend Trieschnigg, Djoerd Hiemstra, and Theo Meder, "Supporting the Exploration of Online Cultural Heritage Collections: The Case of the Dutch Folktale Database," *Digital Humanities Quarterly* 11, no. 4 (2017), accessed May 5, 2018, http://www.digitalhumanities.org/dhq/vol/11/4/000327/000327.html.

however, is to look at other examples which implement processes relevant to your own project. You should consider what makes a particular website or application user-friendly and how it supports complex input.

The technicalities of building the interface are beyond the scope of this handbook, but a certain amount of knowledge can also be gained from existing interfaces. Most research sites will describe how the site is built and additional information can be gleaned from looking at the source code, which will normally reveal what frameworks and libraries are used by the site to structure the interface and create interactive elements.

References

Codd, E. F. "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM* 13, no. 6 (1970): 377–387.

Knirk, James, Helle Degnbol, Bent Chr. Jacobsen, Eva Rode, Christopher Sanders, and þorbjörg Helgadóttir, eds. *A Dictionary of Old Norse Prose/ Ordbog over det norrøne prosasprog*. Copenhagen: The Arnamagnæan Kommission, 1989.

Muiser, I., M. Theune, R. de Jong, N. Smink, R. B. Trieschnigg, D. Hiemstra, and T. Meder. "Supporting the Exploration of Online Cultural Heritage Collections: The Case of the Dutch Folktale Database." *Digital Humanities Quarterly* 11, no. 4 (2017). Accessed May 5, 2018. http://www.digitalhumanities.org/dhq/vol/11/4/000327/000327.html.

Whitelaw, Mitchell. "Generous Interfaces for Digital Cultural Collections." *Digital Humanities Quarterly* 9, no. 1 (2015). Accessed May 5, 2018. http://www.digitalhumanities.org/dhq/vol/9/1/000205/000205.html.

Wills, Tarrin. "Relational Data Modelling of Textual Corpora: The Skaldic Project and Its Extensions." *Digital Scholarship in the Humanities* 30, no. 2 (2015): 294–313.