STAPLE Documentation

Erin M. Buchanan

Invalid Date

Table of contents

1 Preface

Scientific research has become increasingly complex, requiring specialized skills, interdisciplinary work, and collaboration among large teams. Managing such projects and tracking data and metadata has become a significant challenge. The need for FAIR (findable, accessible, interoperable, and reusable Wilkinson et al. 2016) open data, materials, and metadata has similarly grown, especially considering recent mandates for required sharing for federally funded projects (Rep. Ryan 2019; Rep. Foxx 2019). While user-friendly software tools that help researchers structure metadata exist, few researchers are aware of their necessity and usefulness, the onus is still on the project team to collect and curate this information. Studies using teams of researchers require sophisticated tracking of all project elements, and therefore, there is a need for scientific project management software that is tailored to the management of all manner of science projects. Software that promotes best practices in FAIR data and metadata would increase openness in all reporting, and tracking, and reporting standards.

Currently, the tools and websites designed for researchers are focused on *getting* researchers to share their materials, code, and data (i.e., Open Science Framework, FigShare, Zenodo. These repositories represent a necessary resource for long-term storage of outputs, but do not help researchers organize or track information during the life of a study. Project management software, like Asana, Monday, or ClickUp, are designed from a business perspective that is not tailored to scientific research. While the features that scientific research requires may be found in some individual project management software, no current solution provides all the essential features, such as the ability to assign tasks at different scales (e.g., teams, individuals), integrated metadata, fully transparent access to all information in the project manager, and long-term storage that complies with international data privacy regulations. Project management software is designed to *get things done* rather than *document* the way a project was completed, so attempts to use existing software for this often involve 'hacking' it to extract necessary records.

To this end, we developed STAPLE, which not only helps with the unique challenges of project management of research but includes open and transparent documentation of data and metadata, as well as providing templates for minimum metadata collection. STAPLE allows users to add project components based on research type, assign timelines, delegate tasks to individuals or groups, and link to long-term storage of research outputs. STAPLE helps track authorship credit and contributor roles via tasks or contributors. STAPLE includes default metadata standards for documenting common research outputs and is fully open-source, enabling community input and adaptation. Designed for global reach, STAPLE supports the transition to open science by including researchers and outputs that are often overlooked, regardless of

technical expertise. Through its point-and-click interface, and downloadable documentation outputs, STAPLE makes it easier for all researchers to participate in transparent, inclusive, and reproducible science.

term	definition
FAIR	FAIR stands for Findable, Accessible, Interoperable, and Reusable, a set of guiding principles tha
metadata	Metadata is information that describes other data. It helps explain what a dataset or research ou

2 Introduction

The STAPLE mission: *Project Management* software that allows you to *document* your research project to improve *transparency*.

As researchers, we need to focus on the big picture - the overarching goals of a project, timelines, theory, hypotheses, and more. We then need to understand the smaller goals and tasks necessary to achieve the big picture. Researchers are often taught how to run projects, collect data, analyze that data, and write a paper summary. Training in project management is learned by example through the research lab, depending on the supervisor, and often not directly taught. Each project often involves special needs and circumstances, and the addition of even one more local collaborator can create complexity with multiple moving parts.

Research shows that the number of co-authors and collaborators on research projects is increasing over time. The globalization of research, the Internet, and software tools for collaboration (Zoom, GitHub, Google, etc.) has improved our ability to work together on new ideas. Large-scale collaboration has been encouraged to improve the diversity of participants, researchers, and cultures represented in data. Once we add other people (with different lab cultures and expectations!), institutions, geopolitical regions, and languages - the complexity of a research project also increases.

So why do we need software?

Project management software exists: Asana, Monday, Clickup, and Trello for example. These products were designs based on project management templates that are often business focuses. Research projects have their own unique and special needs that do not fit neatly into business models. Large-scale teams have found success treating a research project with many smaller teams as a classroom - but software designed for educational purposes is also not completely compatible with research project goals and management. STAPLE is designed for scientists by scientists with input from important stakeholders including funders, librarians, journals, and more.

So what's the deal with documentation?

In tandem with the encouragement for interdisciplinary, diverse teams, researchers have begun to focus on the improvement of science through transparency and sharing. The Transparency and Openness guidelines (https://www.cos.io/initiatives/top-guidelines) establish ideals for projects to be reproducible, open, and transparency through data sharing, code/analysis sharing, materials sharing, pre-registration, and replication. Journals require different levels of sharing for publication, which has pushed researchers to begin to implement these practices.

However, the simple sharing of data, code, and materials does not make them FAIR: Findable, Accessible, Interoperable, and Reusable. Simple sharing of data does not necessarily allow someone to find that data in relation to the research project, or allow a researcher to know what `V1` means in the data, or even reuse part of the data.

Creating adequate documentation for the reuse of any output from a project can be difficult. There are often no clear standards, the work can be time consuming, and is often left to the end of a project when important details may have been forgotten. STAPLE is designed to encourage you to document information along the way and to provide guidance with what you should include with each document. As you complete specific task, such as create your materials for a study, you will be able to link your project to those materials and include information to interpret and understand those documents. We provide a set of minimum documentation standards for multiple types of documents - and the ability for you to include extra information if your field has set standards.

Ok, so that's transparent?

STAPLE allows you to track and run your project - and then include research outputs from that project. Each step of the way, STAPLE collects data about who did what and when. If you are the only person working on a project, it's simple: you will export a final project timeline that includes information about each step of the project, each output, and the documentation needed to make those outputs useful. If you work with other people, STAPLE will allow you to assign tasks to those individuals (or teams of individuals!) to complete as part of the project management functionality. At the same time, STAPLE tracks their work and creates a project timeline that shows that they completed a task for the project.

This component of the software was designed to align with contributorship models of research - instead of "authorship", each person receives credit for the components they contributed to the project. Each field has different standards for authorship, and many individual's work may be unacknowledge due to those cultures. Using STAPLE, their work would be credited, even if they did not earn final "authorship" on a publication. This output is also useful for defining contributions into systems such as CRediT (https://credit.niso.org/) - where each person's role is binned into categories. By linking each role to specific tasks, we can transparently show what each role means to a specific project.

3 Documentation: Metadata

3.1 What is metadata?

Metadata is *data* or information about a resource. Here, we use the term *resource* to denote anything you want to share with the larger community. Most obvious resources from scientific projects include datasets from which analyses and research reports are derived. However, there are many types of resources that can be shared from a project. For example, we have identified some of the following resources (not an exhaustive list):

- Documents:
  - Ethical approval documents
  - Stimuli and materials
  - Translations of stimuli
  - Other documentation of the research project (notes, approvals, presentations)
  - Manuscript or research report
- Data:
  - Primary: Comes in many forms but likely text, audio, video, or images, created during the project
  - Secondary: information about using another data source
- Code:
  - Code for analysis, study creation, or study implementation
  - Software code

Additionally, metadata can be provided about the global project, the people involved in that project, and how the project was completed:

- Project information
- Author information
  - Organization/Institutions
  - Funders
- Contributions: who did what on the project

3.2 Why should I create metadata?

The purpose of metadata within the scientific community is multi-fold and often described with the acronym FAIR (Wilkinson et al., 2016):

- *Findable*: allowing resources to be findable via database, search engine, etc. Metadata is indexed by journals, libraries, databases, and search engines to assist in discovery.
- *Accessible*: allowing resources to be accessible by providing information on how to download/use resources

- *Interoperable*: the ability to integrate resources with other resources
- *Reusable*: enough information should be provided that resources can be reused by other individuals.

Learn more about FAIR principles here: https://www.go-fair.org/fair-principles/

Metadata can be time consuming, confusing, and hard to make. So, why should you create metadata that makes your work FAIR?

- It can be required:
  - Recent laws across the globe require at least open data to be provided with publications or research reports.
  - Journals have begun to adopt the Transparency and Openness Guidelines (https://www.cos.io/initiatives/top-guidelines), which requires the sharing of resources for a project. For stricter TOP guidelines, metadata would be required to adhere to those standards.
- Reproducibility:
  - Researchers can independently verify your work and correct errors if found (Piwowar & Vision, 2013).
- Extension/Generalization:
  - Researchers can extend your work by using resources in new projects, reanalyzing data in new ways, or combining resources for meta-analysis (Vadillo, Gold, & Osman, 2018; the Human Connectome Project—Van Essen et al., 2013).
- Visibility and Metrics:
  - Publications with open resources have higher citation rates (McKiernan et al., 2016).s

3.3 How metadata is structured

Metadata can fall into two categories:

- Human readable: human readable formats are structured as a report with titles of metadata information and the description for each provided.
- Examples of human-readable formats:
  - Descriptive reports
  - Simple tables provided resources and their descriptions
- Machine readable: provides the same information as the human readable format, but in a structured/standardized format that can be integrated and read by computer systems.
  - Examples of machine-readable formats:
    * Bar codes
    * eXtensible Markup Language (XML)
    * JavaScript Object Notation (JSON)
    * Resource Description Framework (RDF)

In STAPLE, we allow you to create metadata through simple HTML forms that conforms to either our proposed minimum standard or your own uploaded standard. If you have never worked with metadata or created it before, you can use our templates to understand what needs to be provided and when. If your research field has existing standards, you can use STAPLE to ensure you include the required information.

We will describe our minimum metadata standards below and provide examples `JSON` or `JSON-LD` format. JSON is one of most popular machine readable formats because it is "lightweight" as a simple text file of key-value pairs. It resembles a dictionary where each entry has a name and a description.

For example, here's how you might format author information:

```
{
  "author":[
    {"firstName":"Erin", "lastName":"Buchanan"},
    {"firstName":"Marton", "lastName":"Kovacs"},
    {"firstName":"Engerst", "lastName":"Yedra"}
  ]
}
```

The author metadata includes three entries, which have more metadata: first and last names. Generally, metadata is *hierarchical* or *nested* to show the relationship between items (i.e., authors have first and last names). We will create output from STAPLE in JSON-Linked

Data (JSON-LD) format which was created by the Resource Descriptive Framework (RDF Core Working Group, 2004). The LD formatting adds special pieces of information `@context` and `@type` - denoting the specific metadata style you are using and typology of the information you are sharing. This standardization of metadata allows for indexing in databases and search engines (especially https://datasetsearch.research.google.com/).

Here's an example of the English Lexicon Project (Balota et al., 2007):

```
{
  "@context": ["https://schema.org/"],
  "@type": ["Dataset"],
  "name": ["The English Lexicon Project"],
  "fileFormat": [".csv"],
  "contentUrl": ["https://elexicon.wustl.edu/index.html"]
}
```

STAPLE generally uses schema.org naming conventions for our minimum metadata standards. Schema.org is a collaborative working group that has provided a standardization for metadata vocabulary. For example, the author of this documentation may be formatted as:

```
{
  "author":[
    {
      "@type":["Person"],
      "identifier":["https://orcid.org/0000-0002-9689-4189"],
      "givenName":["Erin"],
      "familyName":["Buchanan"],
      "email":["ebuchanan@harrisburgu.edu"],
      "affiliation":["Harrisburg University of Science and Technology"]
    }
  ]
}
```

Many communities use Schema.org and extend their formats for their own individual needs. See for example:

- Bioschemas for the lifesciences (https://bioschemas.org/)
- Brain Imaging Data Structure (BIDS; Gorgolewski et al., 2016; https://bids-specification.readthedocs.io/en/stable/appendices/schema.html) for brain imaging data
- psych-DS (Kline, 2018) for the psychology community
- Finance and business (https://www.w3.org/community/fibo/)

- Automotive industry (http://www.automotive-ontology.org/)

The advantage to using Schema.org is it's popularity - many people use it and computers know how to read it. Further, it is considered *semantic* which details the connections between items (https://en.wikipedia.org/wiki/Semantic_Web).

3.4 STAPLE Metadata

Below, we explore the types of metadata that are available in STAPLE and their minimum requirements. Each of these requirements can be extended (see building your own metadata [LINK]), developed from scratch, or other community requirements can be imported into STAPLE (see converting metadata into STAPLE format [LINK]).

You can view our JSON validation schema by examining each file LINK. JSON validation schema are used to create the forms you see in the app and to ensure that the output files are formatted correctly.

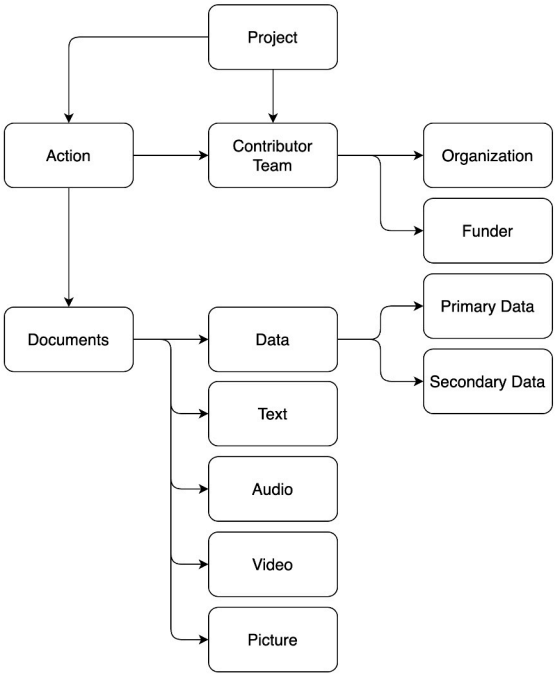
You can view our JSON-LD examples of the output from each schema LINK. These files would be downloaded from STAPLE to share in a repository like GitHub or the Open Science Framework.

The image below shows how each type of metadata collected within STAPLE for our minimum standard is related to each other. Each piece is described below with the required components. Within the descriptions below, we denote the official `keyword` for a object and then the description of what that keyword indicates. These are taken from Schema.org. In the app, they are given more descriptive titles/names to know what to enter in the form.

For example:

- `identifier` –> "Funding number or other identifier:"
- `addressLocality` –> "City name of the Organization's Location:"

```
knitr::include_graphics("pics/metadata.jpg")
```



3.4.1 Project

Projects are separate research endeavors that you want to create information for and track using STAPLE's features. Projects can be as simple as a manuscript/report/document you want to share, grant applications, data collection, stimuli and translations, etc. Generally, we think of projects as work that will eventually be published, but this is not a necessary requirement.

Required elements of project-level metadata (you enter this information):

- `name`: Name of the project
- `description`: Project description
- `keywords`: Keywords
- `citation`: Citation (should probably break this down into parts)

Suggested elements of project-level metadata (you enter this information):

- `abstract`: Abstract of the project, generally from the paper or more formal than the required description
- `identifier`: Persistent identifier (DOI, ISSN, etc.)
- `publisher`: Publisher (should this be under citation)

Other elements of project-level metadata (that are added automatically):

- Contributors
- Corresponding Author
- Funders
- Date

3.4.2 Action

Action describes all of the tasks completed within STAPLE. This metadata is collected for you by using the app. The following information is collected:

- `startTime`: The date-time of when the action was assigned if it was a task
- `endTime`: The date-time of when the action was completed
- `agent`: The person who assigned the task
- `participant`: The person who completed the task
- `result`:
  - `name`: The name of the task
  - `description`: The description of the task
  - `subjectOf`: The project title from staple, linked to the project metadata

3.4.3 Data

Data gets a special type of schema, etc.

https://schema.org/Dataset

3.4.3.1 Primary Data

3.4.3.2 Secondary Data

3.4.4 Project Outputs

Note that data can be text, but this is the spot we classify other objects you may output.

3.4.4.1 Text

https://schema.org/TextObject

3.4.4.2 Audio

https://schema.org/AudioObject

3.4.4.3 Video

https://schema.org/VideoObject

3.4.5 Image

https://schema.org/ImageObject

3.4.6 Author

https://schema.org/Person

3.4.7 Organization

3.4.8 Funder

3.4.9 CRediT

3.5 Add your specific requirements

json format

metadata builder?

4 Overview of STAPLE

4.1 Definitions of Terms

4.1.1 User

A user in STAPLE is a person who has an account and agreed to terms and conditions. Users can create all the elements described below on their own accounts and complete tasks as part of projects that were created an another person's account.

Users have the ability to log in/log out, update their personal information (username, password, email, and profile information), and delete their account. Users also have the ability to add and manage contributors who are part of their projects. They can create teams for individual projects.

Note: Deleting your account *does not* delete you from contributions tied to other projects. You will lose the ability to edit those contributions but you must have the administrator of that project delete you from that project if you wish.

4.1.1.1 Project Admin

A use can be a project administrator:

- By default, those who "create" a project are a project administrator on that project. See the permissions section below for an explanation of what that means.
- Additional project administrators can be added to individual projects.

4.1.1.2 Contributor

- A user can also be added as a contributor on a project. Contributors have limited access to other projects, generally only able to complete tasks that have been assigned to them.

4.1.1.3 Teams

- Teams are a special type of contributor for projects. Each user can be part of multiple teams within a project. Teams are used for tasks that involve a group of people who need to complete a goal. For example, a data collection team for a project with human subjects will need to complete an ethics application to cover their team for data collection. Therefore, each person gets credit for completing the ethics review, but only one person on the team needs to complete the "task" within STAPLE. Another example could be a team of translators who work on converting materials into Spanish from French. The goal is to provide the final Spanish materials, but the entire team gets credit for completing this task.

4.1.2 Permissions

- Project administrators have permissions to edit everything within a project (people, structure [see below], tasks, documentation, and more). They can add, edit, delete, and update all information within a project.
- Contributors only have permission to view and edit tasks that are assigned to them, along with the global project information on a project to which they have been added.

4.1.3 Projects

- Projects are the main organizational structure within STAPLE. Projects are defined by the user but can be thought of as: a research paper, a thesis, a grant application, a meta-analysis, a team report, and more.
- Users will be able to create, read, update, and delete projects. They can define project roles by adding contributors (and contributor teams) and defining additional project administrators. They can create, update, and delete all the possible options described below (elements, tasks, metadata).

4.1.4 Tasks

- Tasks are the "to-do" list within a project. Tasks *must* be assigned to a project. Each task must have a contributor and/or team assigned to them (and the default is the person who creates the task). Additionally, tasks can include a requirement for documentation. These documentation standards can be selected from our minimum requirement list or can your own options can be uploaded.

- The documentation will appear as a "form" online for a person to fill out. They will fill in the required information for that type of task (defined by the creator, please see metadata section for types). These form outputs can be updated by the project administrator at any time or can be updated as part of the review process.
- Once a task has been assigned to a user, the user will get an email to complete the task. At this point, once the user has "completed" the task, the project administrator will receive an email that the task has been completed. They can review the task to ensure it has been completed correctly. If not, they can send the task back to the original user who can then update the task information. This structure is designed in a similar way to GitHub - once a person completes as task, it is reviewed. If ok, the task is marked done, and if not, it requires the original contributor to update and resubmit.

4.1.5 Elements

- Elements are a special placeholder within a project. You can create projects entirely without elements. However, many people like to group tasks into larger categories. Elements are storage boxes for groups of tasks or outputs that fit together. For example, you may create an element for a project that contains all the stimuli materials. This element could be assigned tasks such as "upload the stimuli online", "translate the stimuli into French", "translate the stimuli in Arabic", and so on.
- You can also use elements to label specific sections of a project. For example, you can use elements to hold materials, data, analyses, and the final manuscript, which will put these outputs into labeled sections on the final documentation.
- Tasks can be assigned to elements. By assigning a task to an element, you are grouping them together on STAPLE and the final output documentation provided for transparency purposes.

4.1.6 Dashboard

- The project dashboard allows you to review the elements, tasks, contributors, and documentation available for the project.
- Additionally, you can view and export a human and machine readable timeline for the project. This documentation will include the project documentation collected from contributors, the element structure of the project, and what each contributor did for the project (and a timestamp of when they did it).

5 Example Walkthroughs

In this section, we will examine a few examples of how a user may interact with STAPLE to document their project.

In each of the following examples, you would first log in. These tutorials assume *you* are the data entry manager (project manager, principal investigator, person in charge of managing project tracking). However, data entry for metadata and tasks is the same for project managers and contributors. This section also covers how to enter project information, and thus, does not cover profiles and other small components of STAPLE. Those are described on the STAPLE Elements section. Finally, there are *many* ways to manage the project within the software, and these examples are here to get you started.

To do:

- Implement log in single sign on from ORCID.
- Add glossary terms across this document.
- Update at the end with actual pictures and flow.

5.1 Example 1: One Principal Investigator, Multiple Data Sources

The first example study covers a computational linguistics project in which a researcher collects primary data and uses a secondary data source. In this project, the researcher collects data on semantic priming to show that people read faster when the words are related to each other like *cat-dog* rather than unrelated like *spoon-dog*. Researchers can define similarity - how related words are - in multiple ways, so the researcher wishes to clearly document how they calculated similarity using an secondary data source. Further, the researcher used other psycholinguistic databases to see if they could predict the amount of semantic priming in their study based on the characteristics of the words in the study.

5.1.1 Add a Project

First, you will go to the project page by using the top navigation bar. To add a new project, click "CREATE PROJECT".

Enter your project name and description.

To do: add all project level metadata.

Once the project page is created, you will be brought to the overview page for the project.

To do: the data dashboard is temporary, ideas welcome.

Each component of the project dashboard is explained more on STAPLE Elements section. In short:

- Tasks: a place to create to-dos, track contributions, and add reminders to get things done. Tasks are always tied to projects and a specific user or team.
- Elements: elements are storage boxes for specific project steps. You can think of them as ways to organize your tasks or project. For example, you may use Ethics, Data, Analysis and Results, and Manuscript as ways to group tasks, contributions, and outputs together. Tasks can be tied to a specific element.
- Contributors: people involved in the project. This part does *not* necessarily mean authorship, it simply means someone who did something for the project.
- Settings: the place to update the project level metadata.

To do: figure out how to add users and teams for different levels of project tasks.

5.1.2 Add Elements

I would then add the different elements to my project. I click on elements to get the elements page and create element for adding new areas of organization. There are lots of ways to organize these. I could have created "data" and put both primary and secondary data within that element. I also could add separate elements for the pre-registration (if applicable) or potentially separate materials and translation into two elements.

To do: figure out a view and updating for these.

5.1.3 Add Contributors

From the project dashboard, you can click on contributors to add a new person to the project. Note: this process adds them *only* to that project.

You would be able to add contributors by email on this page. It will notify them that you have added them to the project. Once a contributor has been added to a project, you can begin to assign them tasks.

To do:

- Allow adding by email.
- Create teams and add contributors to them.

5.1.4 Add Tasks

In theory, you could create and run a project with no elements. You would use the task tab for all functionality. To create a task, click on tasks. Note that a lot of task functionality is not available yet, but we will pretend it is.

When creating a task, you will define:

- The task name
- The task description or a how-to complete the task
- The element a task should be assigned to (or none)
- The person or team associated with the task
- If the task needs an "output" (something that needs metadata)
- If so, what type of metadata should the task include?

Once you create a task, the information on what needs to be done will be emailed to the user or team with the task information. They (or you) will be able to complete the task, and information about when the task was assigned and completed will be stored.

If the task has been tagged with needing an output - you will then need to enter the metadata for the requested output. For example, I created a task called "upload IRB documents". In this case, the required output would be ethics approval for the study. I would store that document in my favorite repository (OSF, GitHub, FigShare, etc.) and create a permanent link for the document. Then I would enter the required metadata for that document. In this case, it would be a simple explanation of what the text document contains.

To do:

- Allow tasks to be tagged within element.
- View tasks within elements pages.
- Allow tasks to be tagged with a person or team.
- Allow tasks to be tagged with "needs metadata" or not.
- Provide example of difference in metadata for primary and secondary data.

5.1.5 Project Outputs

To do: Add examples for each type of metadata here. Please see the Metadata section at the moment.

5.1.6 Project Completion

When all tasks are complete and all metadata is entered for the project outputs, you can build a project timeline from the dashboard page. This information will present:

- What was done on a project with a timestamp (example)
  - Who/When a task was assigned
  - Who/When a task was completed
  - Who/When metadata was added
  - Who/When elements were created
  - And so on
- A machine readable format of all the metadata for the study
- A contributorship output matching the model you wish to use
  - For example, tagged CRediT categories

To do: update dashboard to include this option.

5.2 Example 2: Big-Team Project, Multiple Data Collection Sites

6 Installation Guide

SEPARATE: - DEVELOPMENT/CODING - RUN LOCAL MACHINE - PRODUCTION / SERVER VERSION

6.1 Assumptions

- This guide assumes the following:
  - You are working on a Linux server. We are specifically using Ubuntu 20, but this guide should be consistent for most versions of Linux.
  - You have access to the shell command console. You will need access to install files directly onto the server computer.
  - You have worked with the command line before.
  - You have `sudo` rights to add files and services.
  - You know what a hidden file is and at least how to google how to view them.
  - You know what SQL and databases are, even if you've never worked with PostgreSQL before.
  - You know what Apache and/or nginx servers are and how to edit their files.
  - You can at least imagine yourself using git.

6.2 Installation on a Server

6.2.1 Web Server Installation

- Apache - Installation Instructions
- Nginx - Installation Instructions
- They really aren't big fans of each other, only install one. We are using nginx on our server.

6.2.2 Blitz/JS Installation

- Install `node.js` and `npm` - Installation Instructions
  - Other downloads
  - You should have at least node v18+ and npm v9+.
  - You can check your versions by using `node -v` and `npm -v` in a terminal or command window.
  - You may also use yarn, but this guide uses npm.
- Install `blitzjs`: `npm install -g blitz` in a terminal/command window. Check your version is at least v2+ by using `blitz -v` in a command window.

ERIN UPDATE THIS BECAUSE BLAH

- Depending on server setup, you may need `sudo` privileges.

6.2.3 Clone This Repository

- Clone or copy this github repository to the server.
  - Install git on the machine using: `sudo apt-get update sudo apt-get install git`
  - Navigate to `/var/www/html/` or `/var/www/` on the Linux machine.
  - Clone the repository by using this guide - Installation Instructions. `git clone https://github.com/STAPLE-verse/STAPLE.git`
  - This will make a folder called STAPLE with all the files you need.
- Navigate to that folder by using `cd STAPLE`.

6.2.4 Database Installation

- Ensure that you have a local postgres service running on your computer.
- To install see: Installation Instructions.
  - Be sure to write down the superuser information as you are installing the setup for non-Linux machines.
  - You may use other databases, but will need to modify the provided code for their implementation.
- Create the databases for STAPLE. Go to terminal and use:
  - Note that all lines that start with # are comments for explanation.

```
# get into postgres on linux
sudo -u postgres psql
# enter your password for superuser when prompted
CREATE DATABASE staple;
CREATE DATABASE staple_test;
```

CHANGE THE USER TO A DIFFERENT NAME BECAUSE STAPLE CONFUSING

- Create a user with appropriate privileges after postgres installation. While in the terminal and postgres, create a new user:
- Change out `username` and `password` (be sure to leave the quotes!) for your desired user.

```
CREATE USER username WITH PASSWORD 'password';
```

- Get into the STAPLE database. You can use `\l` and should see something like this:

```
postgres=# \l
                                  List of databases
    Name     |  Owner   | Encoding |   Collate   |    Ctype    |     Access privileges
-------------+----------+----------+-------------+-------------+---------------------------
 postgres    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 staple      | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/postgres             +
             |          |          |             |             | postgres=CTc/postgres    +
             |          |          |             |             | staple=CTc/postgres      +
             |          |          |             |             | staple_admin=CTc/postgres
 staple_test | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres              +
             |          |          |             |             | postgres=CTc/postgres
 template1   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres              +
             |          |          |             |             | postgres=CTc/postgres
(5 rows)
```

- Use the following to get into the STAPLE database.

```
postgres=# \c staple
You are now connected to database "staple" as user "postgres".
staple=#
```

- You can check that schema exist using `\dn`:

```
staple=# \dn
 List of schemas
  Name  | Owner
--------+--------
 public | staple
(1 row)

staple=#
```

- Give your user the appropriate permissions to write to the database. Change out `username` for the user you created a minute ago.

```
GRANT ALL ON SCHEMA public TO username;
```

- Use `quit;` to exit out of postgres.

ALTER ROLE your_username CREATEDB; be sure to add this

6.2.5 Connect Database to STAPLE App

- Make sure you are in the folder that you copied the github repository into.
- You can use `ls -al` to view all files in that folder.

```
# for example on my server
erin@staple:~$ cd /var/www/html/STAPLE
erin@staple:/var/www/html/STAPLE$ ls -al
total 764
drwxr-xr-x  13 root root   4096 Oct 24 06:17 .
drwxr-xr-x   6 root root   4096 Oct 24 05:02 ..
drwxr-xr-x   3 root root   4096 Oct 24 04:27 db
-rw-r--r--   1 root root    175 Oct 24 04:27 .editorconfig
-rw-r--r--   1 root root    494 Oct 24 06:17 .env
# more truncated #
```

- Copy the `.env` file and rename it `.env.local`.
  - You may need to turn on settings to see these hidden files on your machine.
  - You can create and edit this file at once with `nano .env.local` or `vim` if you want.
- Ensure the `.env.local` file has required environment variables.
  - After the commented lines, add the `DATABASE_URL` line and change <YOUR_DB_USERNAME> to `username:password` (no <> these are here to show you what to change).

DID NOT RENDER CORRECTLY

```
-   Create and add a `SESSION_SECRET_KEY`.
    -   In the command line prompt, use `openssl rand -hex 16` and copy this long letter/numb
```

```
# This env file should be checked into source control
# This is the place for default values for all environments
# Values in `.env.local` and `.env.production` will override these values
DATABASE_URL=postgresql://<YOUR_DB_USERNAME>@localhost:5432/staple
SESSION_SECRET_KEY=<SESSIONKEY>
```

- Copy the `.env.test` file and rename `.env.test.local`.
- Ensure the `.env.test.local` file has required environment variables in the same way you did above.

```
DATABASE_URL=postgresql://<YOUR_DB_USERNAME>@localhost:5432/staple_test
```

6.2.6 Install STAPLE Requirements

- Make sure you are in the STAPLE main folder. Use the following (not the `#` line, these are notes) to install packages, tailwind, and daisyui.

```
# to install all packages for staple
npm install
# install tailwind css
blitz install tailwind
# install daisyui
npm i -D daisyui@latest
```

- Next, use the following line to create the database structure/schema for STAPLE to run.

```
# to create database with the right set up
blitz prisma generate
blitz prisma migrate dev
```

6.2.7 Starting the App - Local Testing

- In a terminal window, go to the folder you cloned this repository and type:

```
blitz dev
```

- Open (usually) http://localhost:3000 (or whatever it says for localhost in the terminal) with your browser to see the result.
- This step works great on a "regular" computer, but may not be useful for a server. Instead run the service "in production" to view on your website.

6.2.8 Starting the App - Production

- In a terminal window, go to the folder you cloned this repository and type:

```
blitz build
```

- This step may produce errors in the build. You will need to fix these errors before running the application. Check below for common issues.

6.2.9 Keeping the App Going

- Create a service.
- Generally, you might consider putting it here: `/etc/systemd/system/` on a linux machine.
- We've named the file `blitz.service` as an example creating it using `nano`.
- Tutorial for those who do not know how to do this - Installation Instructions

```
# for example
erin@staple:/var/www/html/STAPLE$ cd /etc/systemd/system/
erin@staple:/etc/systemd/system$ nano blitz.service
```

- Example file structure:
- Change the `WorkingDirectory` to your folder.

```
[Unit]
Description=Starts the Blitz service.
After=network.target

[Service]
Type=simple
```

```
WorkingDirectory=/var/www/html/STAPLE
ExecStart=/usr/local/bin/blitz start
Restart=always

[Install]
WantedBy=default.target
```

- Commands:
  - stop: `sudo systemctl stop blitz`
  - start: `sudo systemctl start blitz`
  - restart: `sudo systemctl restart blitz`
  - reload: `sudo systemctl reload blitz`
  - disable: `sudo systemctl disable blitz`
  - re-enable: `sudo systemctl enable blitz`
  - status: `sudo systemctl status blitz`
  - reset: `sudo systemctl reset-failed blitz`

Many thanks to Scott B. for setting this up and giving instructions.

- Run `sudo systemctl start blitz`.
- Check the status using `sudo systemctl status blitz`. Type the letter `q` to exit.
- Status can also help you troubleshoot when you have an error.

6.2.10 Setting Up the Proxy

- Use the following to get to the nginx web server: `cd /etc/nginx/sites-enabled`
- Create a file by using `nano YOURWEBSITE` ... for example ours is `nano app.staple.science` because that is the website of our hosted version of STAPLE.
- Create the server file setup:

```
# Default server configuration
#
server {
    listen 80;
    server_name app.staple.science;
    error_log /var/www/html/YOURFOLDER/logs/web-server.log;
    location / {
        proxy_pass http://localhost:3000/;
    }
}
```

- For `https` you need to set up a certificate and the easiest solution is through `certbot`. See Installation Instructions.

sudo apt install python3-certbot-nginx

6.3 Common Errors

6.4 Run on Your Own Computer

You can also run the software locally on your own computer with a few limitations. If you are using a Linux system, you can follow the instructions above from Blitz/JS Installation through Starting the App - Production. You will be the only user allowed to enter data if you use the software on your own computer, as you are not hosting it on the web for others to connect to. However, you can still use all the functionality of project management and metadata entry/output.

6.4.1 Blitz/JS Installation

- Install `node.js` and `npm` for - Installation Instructions
  - This installer includes both `node.js` and `npm`.
  - You should have at least node v18+ and npm v9+.
  - You can check your versions by using `node -v` and `npm -v` in a terminal or command window.
  - You may also use yarn, but this guide uses npm.
- Install `blitzjs`: `npm install -g blitz` in a terminal/command window. Check your version is at least v2+ by using `blitz -v` in a command window.
- Depending on your computer setup, you may need `sudo` privileges.

6.4.2 Clone This Repository

- Clone or copy this github repository to your computer - Installation Instructions. `git clone https://github.com/STAPLE-verse/STAPLE.git`
- This will make a folder called STAPLE with all the files you need.
- Navigate to that folder in your terminal window.

6.4.3 Database Installation

- Ensure that you have a local postgres service running on your computer.
- To install see: Installation Instructions.

  - Be sure to write down the superuser information as you are installing the setup for non-Linux machines.
  - You may use other databases, but will need to modify the provided code for their implementation.

- Create the databases for STAPLE. Go to terminal and use:
  - Note that all lines that start with # are comments for explanation.

```
# get into postgres on mac
psql -U postgres
# enter your password for superuser when prompted
CREATE DATABASE staple;
CREATE DATABASE staple_test;
```

- Create a user with appropriate privileges after postgres installation. While in the terminal and postgres, create a new user:
- Change out `username` and `password` (be sure to leave the quotes!) for your desired user.

```
CREATE USER username WITH PASSWORD 'password';
```

- Get into the STAPLE database. You can use `\l` and should see something like this:

```
postgres=# \l
                                  List of databases
    Name     |  Owner   | Encoding |   Collate   |    Ctype    |     Access privileges
-------------+----------+----------+-------------+-------------+---------------------------
 postgres    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 staple      | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/postgres             +
             |          |          |             |             | postgres=CTc/postgres    +
             |          |          |             |             | staple=CTc/postgres      +
             |          |          |             |             | staple_admin=CTc/postgres
 staple_test | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres              +
             |          |          |             |             | postgres=CTc/postgres
 template1   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres              +
             |          |          |             |             | postgres=CTc/postgres
(5 rows)
```

- Use the following to get into the STAPLE database.

```
postgres=# \c staple
You are now connected to database "staple" as user "postgres".
staple=#
```

- You can check that schema exist using `\dn`:

```
staple=# \dn
 List of schemas
  Name  | Owner
--------+--------
 public | staple
(1 row)

staple=#
```

- Give your user the appropriate permissions to write to the database. Change out `username` for the user you created a minute ago.

```
GRANT ALL ON SCHEMA public TO username;
```

- Use `quit;` to exit out of postgres.

6.4.4 Connect Database to STAPLE App

- Make sure you are in the folder that you copied the github repository into.
- You can use `ls -al` to view all files in that folder.

```
# for example on my server
erin@staple:~$ cd /var/www/html/STAPLE
erin@staple:/var/www/html/STAPLE$ ls -al
total 764
drwxr-xr-x  13 root root   4096 Oct 24 06:17 .
drwxr-xr-x   6 root root   4096 Oct 24 05:02 ..
drwxr-xr-x   3 root root   4096 Oct 24 04:27 db
-rw-r--r--   1 root root    175 Oct 24 04:27 .editorconfig
-rw-r--r--   1 root root    494 Oct 24 06:17 .env
# more truncated #
```

- Copy the `.env` file and rename it `.env.local`.
  - You may need to turn on settings to see these hidden files on your machine.
  - You can create and edit this file at once with `nano .env.local` or `vim` if you want.
- Ensure the `.env.local` file has required environment variables.
  - After the commented lines, add the `DATABASE_URL` line and change to `username:password` (no <> these are here to show you what to change).

  - Create and add a `SESSION_SECRET_KEY`.
    * In the command line prompt, use `openssl rand -hex 16` and copy this long letter/number combination instead of .

```
# This env file should be checked into source control
# This is the place for default values for all environments
# Values in `.env.local` and `.env.production` will override these values
DATABASE_URL=postgresql://<YOUR_DB_USERNAME>@localhost:5432/staple
SESSION_SECRET_KEY=<SESSIONKEY>
```

- Copy the `.env.test` file and rename `.env.test.local`.
- Ensure the `.env.test.local` file has required environment variables in the same way you did above.

```
DATABASE_URL=postgresql://<YOUR_DB_USERNAME>@localhost:5432/staple_test
```

6.4.5 Install STAPLE Requirements

- Make sure you are in the STAPLE main folder. Use the following (not the `#` line, these are notes) to install packages, tailwind, and daisyui.

```
# to install all packages for staple
npm install
# install tailwind css
blitz install tailwind
# install daisyui
npm i -D daisyui@latest
```

- Next, use the following line to create the database structure/schema for STAPLE to run.

```
# to create database with the right set up
blitz prisma migrate dev
```

6.4.6 Starting the App - Local Testing

- In a terminal window, go to the folder you cloned this repository and type:

```
blitz dev
```

- Open (usually) http://localhost:3000 (or whatever it says for localhost in the terminal) with your browser to see the result.
- Everything you do will be saved this way, so closing the app would mean just closing the app.

6.4.7 Starting the App - Production

- In a terminal window, go to the folder you cloned this repository and type:

```
blitz build
```

- This step may produce errors in the build. You will need to fix these errors before running the application. Check below for common issues.

7 Summary

This section will summarize the main points.

8 Terminology

TBD when glossary is fixed.

9 References

Rep. Foxx, Virginia [R-NC-5. 2019. "H.R.150 - 116th Congress (2019-2020): GREAT Act." https://www.congress.gov/bill/116th-congress/house-bill/150.

Rep. Ryan, Paul D. [R-WI-1. 2019. "H.R.4174 - 115th Congress (2017-2018): Foundations for Evidence-Based Policymaking Act of 2018." https://www.congress.gov/bill/115th-congress/house-bill/4174.

Wilkinson, Mark D., Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, et al. 2016. "The FAIR Guiding Principles for Scientific Data Management and Stewardship." *Scientific Data* 3 (1): 160018. https://doi.org/10.1038/sdata.2016.18.