White Paper
The FieldLinguists' App

1 Project Abstract

The FieldLinguists' App is an OpenSource database that will allow language researchers to securely enter, store, organize, annotate, and share linguistic data. The application will be accessible everywhere; it will run on three different systems (Mac, Linux, and Windows) as a Chrome extension and will be suitable for both online and offline use. Furthermore, the application will be created with collaborative goals in mind; data will be syncable and sharable with other researchers. Researchers can form teams that contribute to a single corpus, where team members can modify and discuss the data from within the application. The system will also have a simple and friendly user interface, allowing participants to drag and drop files (audio, video, text), or record them directly into the database when using the Android app. In addition, the application will have import and export capabilities for multiple file types. Most importantly, the application is designed intuitively and theory free, so it is not necessary to be a field linguist or programmer to figure out how it works.

2 Statement of Need

The FieldLinguists' App is conceived out of the needs of language researchers doing fieldwork or lather large scale data collection. Linguistic fieldwork often requires researchers to travel to places where a stable connection to the internet is not guaranteed. Also, it often involves a group of researchers contributing to building a single database. An ideal linguistic database should therefore work both on- and off-line as well as make it easy to share and integrate data.

There are existing programmes/software used for linguistic fieldwork, however, they fall short in providing features necessary for collaborative fieldwork. For example, some web-based databases (e.g. *Karuk Dictionary and Texts* http://linguistics.berkeley.edu/~karuk/links.php, *The Washo Project* http://washo.uchicago.edu/dictionary/dictionary.php) work only online, hence it is impossible for researchers to enter new data or search the database while in the field where the internet is unavailable.

Non-web-based software such as *Toolbox* (http://www.sil.org/computing/toolbox/) and *FLEx/FieldWorks* (http://fieldworks.sil.org/flex/) are excellent in terms of annotating data and organizing data into various formats (corpus, grammar or lexicon). Nonetheless, integration of data taken by individual researchers is not well-automated and takes considerable work by researchers. Moreover, they run only on a single platform (either PC or Linux, but not both). These tools therefore cause further difficulty in data sharing among researchers using different platforms.

General purpose database software such as *FileMaker Pro* can be customized for the purpose of language research. However, it demands researchers to learn the software, and research teams often need to hire a programmer to customize the software for their research purposes. The stand-alone nature of such software makes data sharing and integration difficult as well.

The existing linguistic database programmes, although useful, have various shortfalls that would hinder collection and integration of data. Some of them are constrained by the internet accessibility or by the computer platform types. Some others demand extra human work in order to integrate data. The present project grows out of discussion with a numbers of fieldworkers dissatisfied with currently available options.

3 App Description

The FieldLinguists' App will enable those interested in language research, preservation, and documentation to securely enter, store, organize, annotate, and share linguistic data. Moreover, the application will be easily customizable to fit specific needs. To accomplish these tasks, the database will be equipped with a variety of features. The following functional requirements are based on a few important considerations where most existing fieldlinguistics/corpus linguistics databases applications fall short.

3.1 Functionality

This application will be able to perform the necessary functions needed by field linguists. The dashboard will be composed of several widgets. The Data Entry widget will be the primary focus, containing four core fields customary for a gloss format (utterance, morpheme-segmentation, gloss, translation). In addition to these, researchers will be able to add customized fields, such as phonetic transcription or context for an utterance. Researchers can even upload audio files and link them to the appropriate data. Each data entry will be tagged for the researcher, date of elicitation and consultant's code.

Furthermore, researchers will be able to add tags for categorization and mark the status of each individual data entry as "Checked" or "To be checked with consultant", which further aids organization and reduces the number of errors that inevitably occur during research. Other functions such as importing data and exporting data into various formats aid efficiency and convenience.

Another widget will be an Activity Feed View displaying the most recent changes. This widget allows researchers to keep up to date on their team's activity. The activity feed will display items such as recent additions to the corpus, comments made on data entries, and recent edits.

Finally, the application will have a powerful search function that will expand into a data list that is contained in its own widget. Data lists can be sorted, saved and can be used for batch operations such as exporting or converting into LaTeX.

This application integrates the best functions from existing fieldwork database programs, while avoiding many of the shortcomings discussed above. The core features are summarized as follows:

- **Modern**
    - **Simple** The system will be designed to replace Word Documents or LaTeX documents which is a very common way field linguists store data because it requires no training, doesn't require a complicated set-up for data categories, and takes no time to add new categories.
    - **Attractive** The system will have a modern design like many of the popular websites such as Google and be customizable so that the user can include a picture of where they are doing research as a background.
- **Powerful**
    - **Smart.** The application will guess what users do most often, and automate the process for them. Most importantly, the system will have automated predictable glossing information.
    - **Searchable.** The application will be designed for search as this is one of the most fundamental tasks a language researcher must be able to do. The search will go far beyond traditional string matches and database indexes; it will be able to display data in context.
- **Data-Centric**
    - **Atheoretical.** The application will not include categories or linguistic frameworks or theoretical constructs that must be tied to the data. The application will allow an analysis to develop organically as data collection proceeds, as opposed to imposing a particular construct upon entry. Researchers will be able to set and change their own categories for the data whenever they choose to.

  - **Collaborative.** The system will have users and teams, and permissions for corpora. Permissions will ensure that data can be safely shared and edited by multiple users. Moreover, the corpus will be versioned so that users can track changes and revert mistakes.
  - **Sharable.** The application will allow researchers to share their data with anyone interested in their work. The application will be able to import data from ELAN XML, CSV and text file formats, and to export to XML, Latex and WIki formats. These import/export functions will make it easier to exchange data with those who are not using the same application.

- **Accessible**
  - **Cross-Platform.** The application will be available for any device that runs a browser. Specifically, the application will be installable as a Chrome extension on Mac, Linux, and Windows computers, as well as Android tablets and phones, and will run both online and offline on these devices. The application will run online on iPads and iPhones.
  - **Portable.** Touch tablets are one of the easiest tools to carry and use in field; they have a long battery life; they can play videos or show images for the consultant to elicit complicated contexts; and they permit recording audio and video and direct publishing to YouTube and/or other services. Furthermore, Android tablets are particularly easy to program and integrate the microphone directly into the database.
  - **Work offine.** Running a webapp offline will have considerable consequences for how data is stored, how data is retrieved, and how much data can be used while offline. Most browsers have limits on the amount of data a webapp can store offline. By delivering a version of the application in a Chrome extension, which has permission to have unlimited storage, researchers will be able to have a significant portion of their data at their fingertips, regardless of the location.
  - **Multiple storage** Data will be stored on a CouchDB server, and will be accessible and sharable by multiple users. The installable Chrome extension version of the application allows individual researchers to store a portion of a corpus, or an entire corpus, on their own devices, enabling offline work and quick search.

- **Open**
  - **OpenData**. Corpora often contain sensitive information, informant stories and other information which must be kept confidential. Having confidential data in plain text in a corpus forces the entire corpus to be kept confidential.

Instead, the system will encrypt confidential data and store the data in the corpus encrypted. To access the plain text the user will have to log in and use a password to decrypt the data. This design has important ramifications for exporting data, and for editing the data outside the application.

- **OpenSource**. Being OpenSource allows departments to install and customize the database application to tailor specific needs without worry that the company behind the software will disappear or stop maintaining the software. In addition, OpenSourcing the software on GitHub will allow linguists with scripting or programming experience to contribute back to the software to make it more customized to their needs, language typologies, or linguistics research areas.
- **Unicode**. Encoding problems and losing data should be behind us in the days of unicode. However, many existing fieldlinguistics databases were built in programming languages that did not support unicode, so the unicode support is dangerously fragile.

4 Goals & Objectives

The principal goal of The FieldLinguists' App is to help language researchers collect and organize linguistic data and to facilitate collaborative research work. The main objectives are to provide:

- A self-explanatory, easy-to-use user interface so that researchers can start using the application at the time the installation is completed.
- Both on- and off-line functionality so that the fieldwork is not constrained by the internet accessibility.
- Customizable data entry fields to accommodate particular requirements of a research.
- Data sharing and integration functions to facilitate collaboration among researchers and between researchers and language consultants.

Although it is designed primarily for linguists, the application will equally be useful for researchers documenting endangered languages and/or creating dictionaries/grammar books for minority languages, as well as language teachers creating educational materials.

5 Budget & Timeline

The FieldLinguists' App is composed of eight modules and thus the cost is divided into eight major components. In addition, a separate price is given for software architecture and for 1 year of user support and project growth, which is needed to make a longterm viable and useful tool that field linguists can adopt for their labs or for their field methods courses. The cost is calculated by determining the time in hours and multiplying by $42-58, the average rate of a software developer in Montréal.

Since the budget is dependent on how long it takes to complete the modules, it is possible to focus on each module separately, thus reducing the time of completion.

Module	Weeks	Price
Software Architecture	0.5	$1,555.20
Collaboration Module	2.5	$5,728.32
Corpus Module	4.2	$9,201.60
Web Spider Module	2.5	$2,177.28
Lexicon Module	2.0	$7,340.54
Phonological Search Module	3.0	$2,177.28
Dictionary Module	22.3	$18,781.63
Glosser Module	19.7	$17,770.75
Aligner Module	10.2	$ 9,787.39
User Support	21.1	$30,246.70
TVS and TPQ		$10,833.32
Total	88	$83,176.04

Table 1: Project Summary

5.1 Collaboration Module

Iteration	Hours	Technology
Software Architecture Design	20	Software Engineering
Collaboration API on central server	30	Software Engineering
Users Model	15	Javascript
Informant Model	15	Javascript
Team Model	15	Javascript
Bot Model	15	Javascript
User Activity Model	8	Javascript
Team Feed Widget	25	HTML5
User list item Widget	16	HTML5
Team Preferences Widget	8	HTML5
User Profile Widget	8	HTML5
User Tests	30	Javascript
Informant Tests	30	Javascript
Team Tests	30	Javascript
Android Deployment	15	Java
Chrome Extension Deployment	20	Javascript
Heroku Deployment	5	Integration

Table 2: The Collaboration Module is used to permit collaboration with teams and users.

5.2 Corpus Module

Iteration	Hours	Technology
Software Architecture Design	20	Software Engineering
Corpus API on corpus server	20	Software Engineering
Corpus Model	8	Javascript
Session Model	8	Javascript
Datum Model	8	Javascript
Datum status model	8	Javascript
DataList Model	8	Javascript
Confidential datum encrypter	16	Javascript
Audio upload and play logic	8	Javascript
Corpus DB implementation on Android	20	Java
Corpus DB implementation on Chrome	20	Javascript
Corpus DB implementation on Node.js	20	Javascript
Corpus versioning Logic	25	Javascript
Corpus Preferences Widget	6	HTML5
Session Preferences Widget	6	HTML5
Datum Preferences Widget	20	HTML5
Datum Status Preferences Widget	16	HTML5
DataList Preferences Widget	6	HTML5
Corpus sync logic	10	Javascript
Corpus diff Widget (to show before sync)	10	HTML5
Insert Unicode Character Widget	10	HTML5
Corpus Details Widget	6	HTML5
Session Details Widget	6	HTML5
Datum Details Widget	20	Javascript
DataList Widget	30	Javascript
Global Search logic	30	Javascript
Power Search logic	80	Javascript
Corpus Tests	5	Javascript
Session Tests	10	Javascript
Datum Tests	10	Javascript
Datum Status Tests	10	Javascript
DataList Tests	20	Javascript
Heroku Deployment	5	Integration

Table 3: The Corpus Module is used to sync, share, edit, tag, categorize and open data.

5.3 Web Spider Module

Iteration	Hours	Technology
Corpus Visualization Widget	40	HTML5
Web Spider Training Logic	60	Java

Table 4: This module is a subportion of the Corpus Module.

5.4 Lexicon Module

Iteration	Hours	Technology
Software Architecture Design	20	Software Engineering
Lexicon API on Lexicon server	20	Software Engineering
Lexicon Model	6	Javascript
Morpheme Model	6	Javascript
Allomorph Model	6	Javascript
Gloss Model	6	Javascript
Orthography Model	16	Javascript
Lexicon DB implementation on Android	20	Java
Lexicon DB implementation on Chrome	20	Javascript
Lexicon DB implementation on Node.js	20	Javascript
Lexicon versioning Logic	10	Javascript
Lexicon Preferences Widget	6	HTML5
Morpheme Tests	6	Javascript
Allomorph Tests	6	Javascript
Gloss Tests	6	Javascript
Orthography Tests	8	Javascript
Lexicon Analysis Widget	10	HTML5
Lexicon sync logic	10	Javascript
Lexicon diff Widget (to show before sync)	10	HTML5
Lexicon Details Widget	6	HTML5
Lexicon Tests	12	Javascript
Heroku Deployment	5	Integration

Table 5: The Lexicon Module is used to house, and read lexicon entries to be used for the glosser.

5.5 Phonological Search Module

Iteration	Hours	Technology
Phonology Ontology for phonological search	60	Java
Lexicon Visualization Widget	40	Javascript
Lexicon Editing Widget	20	Javascript

Table 6: This module is a subportion of the Lexicon Module.

5.6 Dictionary Module

Iteration	Hours	Technology
Software Architecture Design	40	Software Engineering
Dictionary API on Lexicon server	30	Software Engineering
Semantic Model	60	Javascript
Syntactic Model	60	Javascript
Citation Model	60	Javascript
Synonyms Model	60	Javascript
Dictionary DB implementation	80	Integration
Dictionary Training Logic	80	Java
Web Spider Training Logic	100	Java
Dictionary Preferences Widget	8	HTML5
Dictionary WordNet Analysis Widget	120	HTML6
Dialect Profile Widget	8	HTML5
Semantic Tests	30	Javascript
Syntactic Tests	30	Javascript
Citation Tests	30	Javascript
Synonyms Tests	30	Javascript
Spider Tests	30	Java
Training Tests	30	Java
Heroku Deployment	5	Integration

Table 7: The Dictionary Module is used to share the lexicon in the form of a WordNet/Wiktionary dictionary with the language community as required by some grants.

5.7 Glosser Module

Iteration	Hours	Technology
Software Architecture Design	40	Software Engineering
Glosser API on Lexicon server	30	Software Engineering
Morpheme Model	15	Javascript
Allomorph Model	15	Javascript
Gloss Model	15	Javascript
Orthography Model	30	Javascript
Glosser DB implementation	80	Integration
Glosser Prediction Logic	80	Java
Glosser Machine Learning Logic	80	Java
Glosser Training Logic	80	Java
Web Spider Training Logic	80	Java
Glosser Preferences Widget	8	HTML5
Morphological Analysis Widget	40	HTML6
Dialect Profile Widget	8	HTML5
Morpheme Tests	30	Javascript
Allomorph Tests	30	Javascript
Gloss Tests	30	Javascript
Orthography Tests	30	Javascript
Spider Tests	30	Java
Training Tests	30	Java
Heroku Deployment	5	Integration

Table 8: The Glosser Module is used to automatically gloss datum, smarter than the standard lexicon.

5.8 Aligner Module

Iteration	Hours	Technology
Software Architecture Design	10	Software Engineering
Aligner API on Lexicon server	10	Software Engineering
Dictionary Model	15	Javascript
Aligner DB implementation	80	Integration
Aligner Machine Learning Integration	80	Java
Aligner Preferences Widget	8	HTML5
Audio Waveform Visualization logic	30	Javascript
Audio Spectrogram Visualization logic	?	Javascript
Transcription User Interface	80	HTML5
TextGrid export	20	Javascript
Dialect Profile Widget	8	HTML5
Orthography Tests	30	Javascript
Training Tests	30	Java
Heroku Deployment	5	Integration

Table 9: The Aligner Module is used to create TextGrids from the orthography and the audio files, used for prosody and phonetic analysis.

5.9 User Support

Iteration	Hours	Technology
Sample data	30	Linguistics
Integrate software with sample data	30	Javascript
Screencasts on how to use the app(s)	24	Quicktime/YouTube
Screencasts on how to modify the code	40	Quicktime/YouTube
Server maintenance	20	Integration
Monitor server costs and develop pricing plan	100	Business
Answer user emails	250	Support
Read twitter feeds and facebook channels	100	Support
Help IT/developers install and set up the server on their department servers	50	Support
Upgrade javascript/android libraries	40	Javascript
Amazon EC2 server CPU+Memory+Bandwidth		Server
Release new versions	160	Javascript

Table 10: User Support includes 1 year of product support and project growth. It is needed to make a longterm viable and useful tool that field linguists can adopt for their labs or for their field methods courses.

6 Evaluation

The usefulness and effectiveness of the FieldLinguists' App will be evaluated against the following criteria, taken and modified from E-MELD Best Practices in Digital Language Documentation (http://emeld.org/school/what.html) and DataONE Primer on Data Management (http://www.dataone.org/sites/all/documents/DataONE_BP_Primer_020212.pdf).

- Collection: Data are collected and organized with ease.
  Self-explanatory UI and on- and off-line functionality will make data collection easier.

- Quality: The quality of data is assured.
  Datum status and comments functions enables researchers to check and inspect the data quality.

- Description: Data are annotated and categorized according to various terminological conventions.
  Data fields and tags accommodate metadata categories that researchers choose to use. The application is not tied to a particular terminological convention.

- Integration: Data from disparate sources are combined into one consistent data set.
  Sync function helps integrate data taken by multiple researchers into one corpus. Import/export functions enables conversion of data format, making data sharing easier.

- Analysis: The application organizes data in ways to help data analysis.
  Customizable data entry fields and data tags, as well as search/list view functions allow researchers to organize data ready for analysis.

- Format: Data are intelligible regardless of the types of operating system.
  The application runs on PC, Linux and Mac OS as a Chrome extension, as well as on Android. Data supports Unicode and exportable to XML files.

- Discovery: Data are searchable and discoverable.
  The application is accessible through general internet search. Within the application, data are discoverable via keyword search. The two-step data discovery process will minimize irrelevant search results.

- Access: Data are accessible.
  Data stored in the application are open to public view in principle, however authorized researchers (authors of data) have control over who can see and/or edit their data.

- Citation: Data provide citation information.
  Data source will be cited with the name of the corpus in fieldlinguists.com (e.g. fieldlinguists.com#corpus/corpus-name).

- Preservation: Data are archived in a way that withstands long-term preservation.
  Data are stored in a host server as JSON files. JSON files are human-readable text files, therefore the information content would not be lost even if the data format becomes obsolete.

- Security: Confidential information must be kept from public access.
  Encryption will ensure that confidential information such as language consultant's identity be hidden from public access unless the consultant decides otherwise.