Project Report - NS Plants Book iOS App

Mitchell Keenan

Project Summary

My project is to create an iOS app which presents the data contained in and mimics the functionality of the Nova Scotia Plants book. I was told that the goal for my project was not a complete app, but to get a well documented start on it including database design, research and project code, as the project is to be completed by a summer student.

The completed app should:

- Be useful and accessible to both the layman and the biologist.
- Provide all relevant information for each species, including photos and location maps.
- Allow the easy identification of plants through the Dichotomous Key.
    - At each level, all plants matching the current description should be viewable.
- Providing the ability to easily find a plant species by various names, and possibly location.
    - This should deal with misspellings and missing diacritic marks reasonably well.
- Be completely functional without an internet connection as often times it will be used in remote locations where cell coverage and wifi are not available.
- Have a searchable glossary of terms which is accessible in any state of the app.
- Have an easily updatable database (web-portal) for data entry/modification which can be pushed to the app during the build pipeline.

The completed app might:

- Let users search for plants by text or gps location.
- Support overlaying current gps location on the mini-maps shown for each species.
- Have a social aspect where users can upload their own photos to a central server.
    - These would be visible when the app had access to the internet.
    - This is a large scope goal, as it involves the app accessing a secondary online database, support for user accounts, moderators, and editing/removing pictures.
- Allow the saving of certain plant species to a ‘favourites’ type list.

Work So Far

So far, the database design and implementation, as well as initial UI/UX design has been completed, including mockups of major screens. Investigations and planning for both the app code and for the web-based DB front-end have been started in earnest. Some more details of the completed work can be found on the second page.

Work To Come

Implementation of a simple layman-usable data entry system connected to the primary database, a build pipeline which copies this database to the app, and as many features of the core app as can be completed. Anything I don’t finish, UI polish, and any extraneous feature support are left to the next student taking over the project.

Technologies and Research

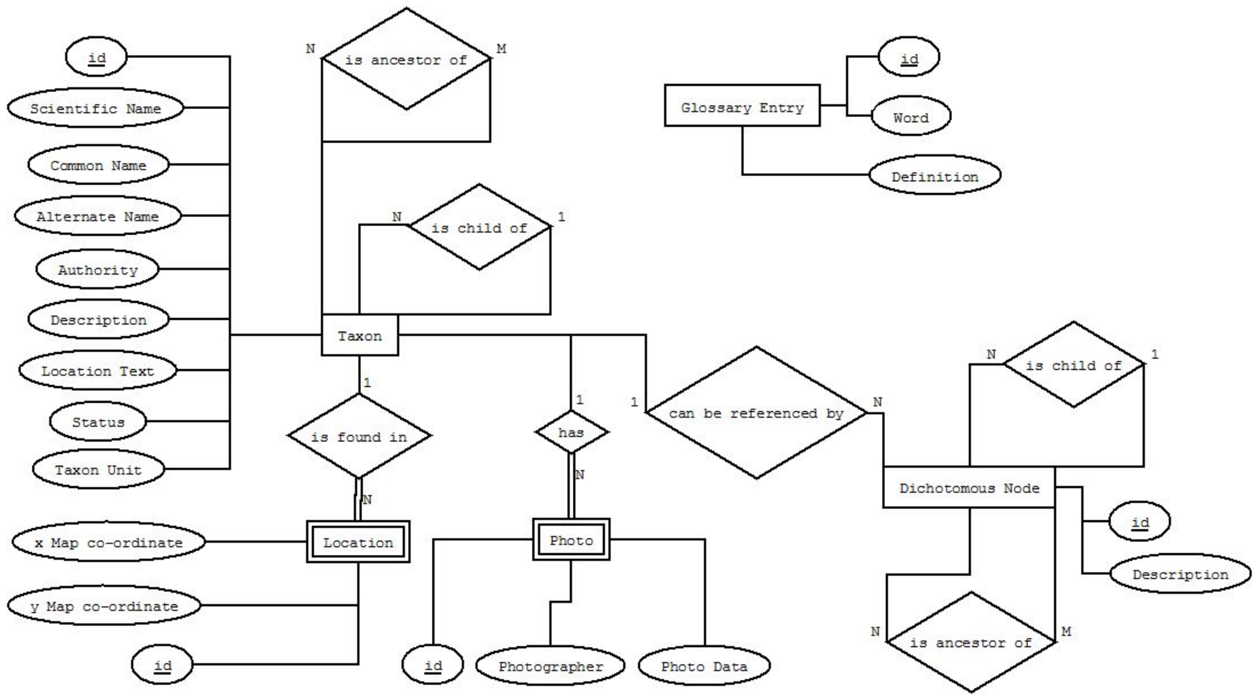
The app is being developed in Apple's swift language and uses an SQLite database. I chose swift as it is the simplest/most convenient language for iOS development right now, I was unfamiliar with swift and iOS development before the project so I have spent a fair amount of time learning the basics. SQLite was chosen because it is a good lightweight database engine for an embedded scenario and will support the data structures needed for the project as well as the ability to have a central database with web-access unlike Apple's.

I also chose to use Stephen Celis' SQLite wrapper for swift. This provides a solid type-safe way of interacting with our SQLite database in swift. It also ensures we are avoiding any SQL compilation issues with our queries.

The data-entry portal will be written in PHP and allow a barebones access to the database, including input forms for each taxon(species and their parents etc.) as well as each step of the dichotomous key. The glossary should be able to be parsed from the book without issue and as such shouldn't need any data-entry.

Database Design

I spent a fair amount of time doing research in how the database for the app should be built. The structure of taxonomic/hierarchical data, along with the types of access we require present somewhat interesting problems in database design, which led to some seemingly odd decisions when viewed without context. Let's take a look at the entity relationship diagram:



You may notice some seemingly unnecessary/duplicate relations for taxon and dichotomous node, however these make sense when you understand their uses. When traversing the dichotomous key or viewing a non-species entry, the user must be able to view the immediate children of the current entry as well as view all entries which fall below the current one in the given hierarchy. To be done efficiently, these two tasks necessitate the 'is child of' and 'is ancestor of' relations. I encountered this problem during my database design and eventually found a well thought-out presentation on the subject by Bill Karwin.

My SQLite database setup script can be found in Appendix A. Of note here is the absence of SQLite virtual tables for full-text search. These are yet to be implemented, and should, when combined with some regex tricks, provide the search capabilities required for the app. If this still is not sufficient a list of common misspellings of the various names of the plants could possibly be generated and compared against when searching.

UI/UX Design

The app will use the 'tabbed interface' template present in xcode and will adhere as strictly as possible to current iOS design patterns. In Appendix B are some simple mockups showing the major screens.

- Note that the menu and it's tab button have been removed in my current design.
- Note that the sliding up and down of the glossary is no longer included in the design. Users should be able to switch to this tab to look something up, then return to their previous state in any other tab.

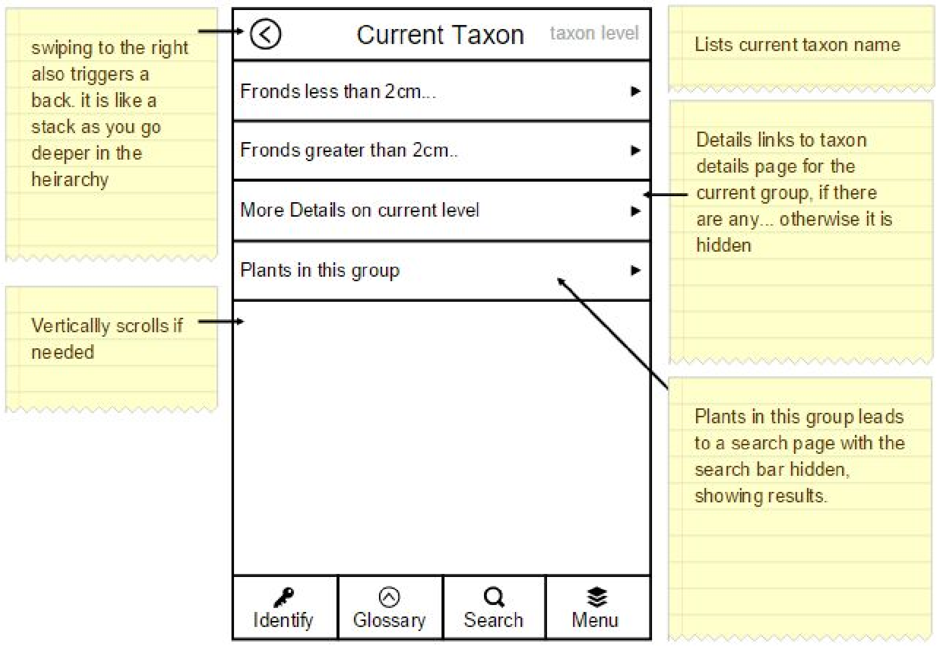
Appendix A - SQLite Table Creation

```
PRAGMA foreign_keys = true;
DROP TABLE IF EXISTS taxon;
CREATE TABLE taxon (
    id          INTEGER PRIMARY KEY AUTOINCREMENT,
    unit_type   INTEGER NOT NULL,
    parent      INTEGER REFERENCES taxon(id),
    sci_name    TEXT DEFAULT '',
    common_name TEXT DEFAULT '',
    alt_name    TEXT DEFAULT '',
    authority   TEXT DEFAULT '',
    description TEXT DEFAULT '',
    loc_text    TEXT DEFAULT '',
    origin      TEXT DEFAULT '',
    status      TEXT DEFAULT ''
);
DROP INDEX IF EXISTS taxon_parent_idx;
CREATE INDEX taxon_parent_idx ON taxon (parent);
DROP INDEX IF EXISTS taxon_unit_type_idx;
CREATE INDEX taxon_unit_type_idx ON taxon (unit_type);

DROP TABLE IF EXISTS taxon_closure;
CREATE TABLE taxon_closure (
    ancestor    INTEGER NOT NULL REFERENCES taxon(id),
    descendant  INTEGER NOT NULL REFERENCES taxon(id),
    PRIMARY KEY(ancestor, descendant)
) WITHOUT ROWID;
DROP INDEX IF EXISTS taxon_ancestor_idx;
CREATE INDEX taxon_ancestor_idx ON taxon_closure (ancestor);
DROP INDEX IF EXISTS taxon_descendant_idx;
CREATE INDEX taxon_descendant_idx ON taxon_closure (descendant);

DROP TABLE IF EXISTS location;
CREATE TABLE location (
    id      INTEGER PRIMARY KEY AUTOINCREMENT,
    taxon   INTEGER NOT NULL REFERENCES taxon(id),
    x       REAL NOT NULL,
    y       REAL NOT NULL
);
DROP INDEX IF EXISTS location_taxon_idx;
CREATE INDEX location_taxon_idx ON location (taxon);
```

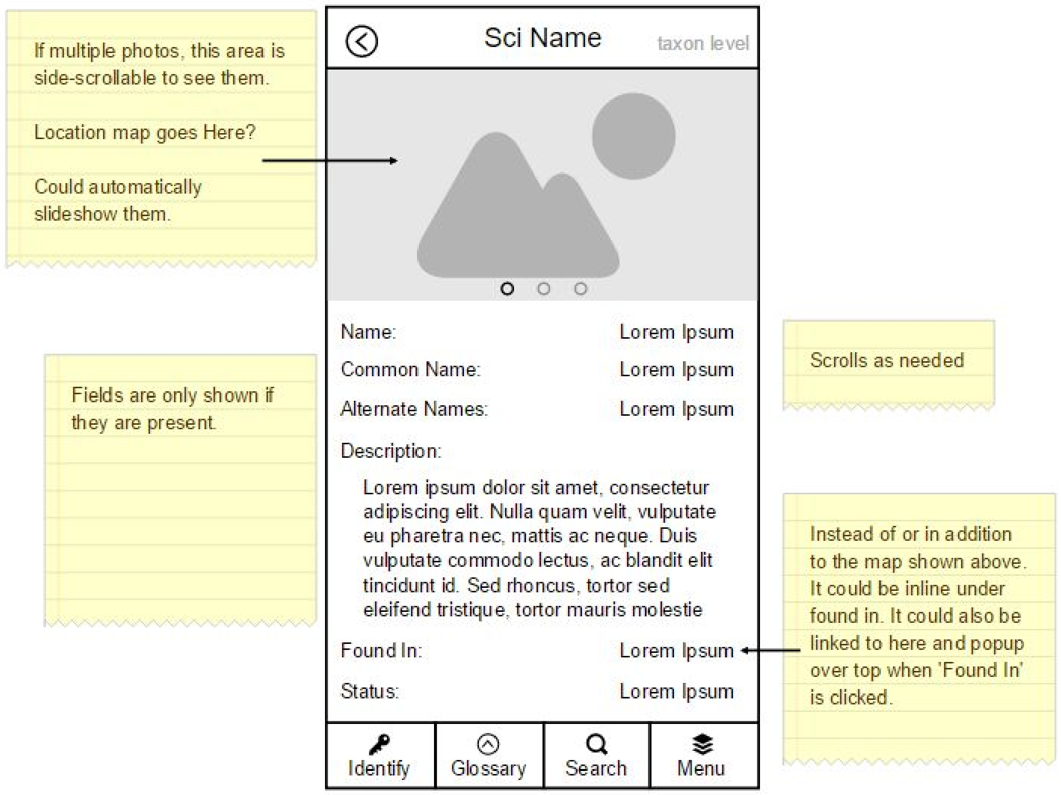
```
DROP TABLE IF EXISTS photo;
CREATE TABLE photo (
    id      INTEGER PRIMARY KEY AUTOINCREMENT,
    taxon   INTEGER NOT NULL REFERENCES taxon(id),
    autor   TEXT DEFAULT '',
    photo   BLOB NOT NULL
);
DROP INDEX IF EXISTS photo_taxon_idx;
CREATE INDEX photo_taxon_idx ON photo (taxon);

DROP TABLE IF EXISTS word;
CREATE TABLE word (
    id          INTEGER PRIMARY KEY AUTOINCREMENT,
    word        TEXT NOT NULL,
    definition  TEXT NOT NULL
);
DROP INDEX IF EXISTS word_idx;
CREATE INDEX word_idx ON word (word);

DROP TABLE IF EXISTS key_node;
CREATE TABLE key_node (
    id          INTEGER PRIMARY KEY AUTOINCREMENT,
    description TEXT NOT NULL,
    parent      INTEGER REFERENCES key_node(id),
    taxon       INTEGER REFERENCES taxon(id)
);
DROP INDEX IF EXISTS key_node_parent_idx;
CREATE INDEX key_node_parent_idx ON key_node (parent);

DROP TABLE IF EXISTS key_node_closure;
CREATE TABLE key_node_closure (
    ancestor    INTEGER NOT NULL REFERENCES key_node(id),
    descendant  INTEGER NOT NULL REFERENCES key_node(id),
    PRIMARY KEY(ancestor, descendant)
) WITHOUT ROWID;
DROP INDEX IF EXISTS key_node_ancestor_idx;
CREATE INDEX key_node_ancestor_idx ON key_node_closure (ancestor);
DROP INDEX IF EXISTS key_node_descendant_idx;
CREATE INDEX key_node_descendant_idx ON key_node_closure
(descendant);
```

Appendix B - Mockups

Dichotomous Key Navigation Mockup


swiping to the right also triggers a back. it is like a stack as you go deeper in the heirarchy
Current Taxon
taxon level
Lists current taxon name
Fronds less than 2cm...
Fronds greater than 2cm..
More Details on current level
Details links to taxon details page for the current group, if there are any... otherwise it is hidden
Plants in this group
Vertically scrolls if needed
Plants in this group leads to a search page with the search bar hidden, showing results.
Identify
Glossary
Search
Menu

View Taxon Mockup


If multiple photos, this area is side-scrollable to see them.
Location map goes Here?
Could automatically slideshow them.
Sci Name
taxon level
Name: Lorem Ipsum
Common Name: Lorem Ipsum
Alternate Names: Lorem Ipsum
Description:
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie
Found In: Lorem Ipsum
Status: Lorem Ipsum
Scrolls as needed
Fields are only shown if they are present.
Instead of or in addition to the map shown above. It could be inline under found in. It could also be linked to here and popup over top when 'Found In' is clicked.
Identify
Glossary
Search
Menu

Search Mockup

Search is similar to glossary, but is a standalone page.

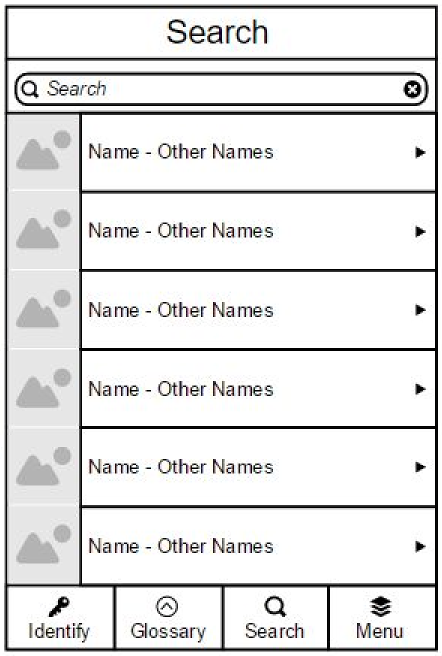
Scientific, Common and Alternate names of any entries (species or otherwise) should be searchable.

Fuzzy-find or some other utility should deal with misspelled or unproperly accented search strings.

Search should start showing all plants.

Typing narrows/filters the results in a live manner.

scrolls



Search bar is greyed out and filled with "Plants in group A" when searching from a taxon with the "Plants in this group option".

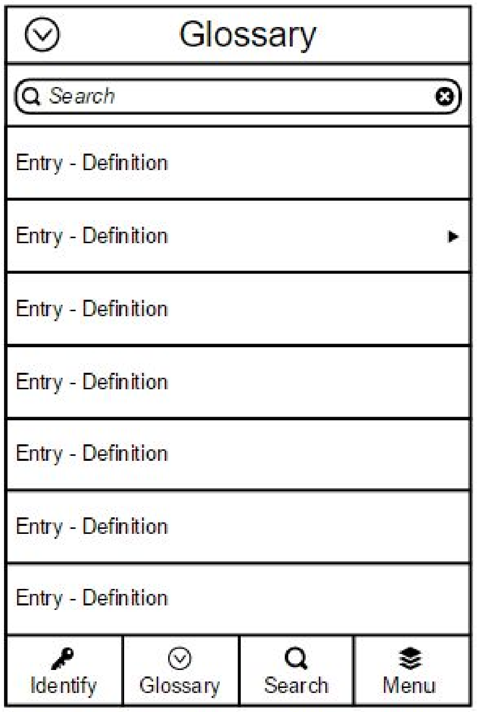
Entries should have the name which matches the search featured, with other names shown.

A picture should be featured on the left

Glossary Mockup

Glossary will be slide up and overlay whatever screen the user is on at the time. The back button is replaced with a down button to slide the glossary back down.

scrolls



Search should either be showing all words by default or possibly any words on the current page which are in the glossary.

Typing narrows/filters the results in a live manner.

Entries should have their definitions on the option. Depending on length or other relevant info they could have a link to another screen - but this is probably unnecessary