Team: Floradex
Amy, Robin, Siyao, Skyler

Summary: Our goal is to develop an iPhone app that will allow users to identify plants through either a dichotomous key or keyword search.

Components

Homepage:

This will give the user the option of accessing the dichotomous key, keyword search, or MyPlants list (i.e., the actual functionality of the program).

Dichotomous Key:

Traditionally, a dichotomous key works like a binary tree--each node contains a question about the plant's characteristics with branches corresponding to the possible answers (since it's dichotomous rather than polychotomous, there will only be two choices). The leaves of the binary tree are the names of the plants with all of the previously indicated characteristics.

From the user's perspective, the interface would display a question (for example, does the tree have broad leaves or needles?), buttons for each option with pictures to clarify what each means, and an "skip" button. Pressing the answer button will take the user to either the next question or the result. Pressing the "skip" button will take the user to the list of plants with the characteristics that have been specified so far.

From a more technical perspective, the dichotomous key will be stored as a binary tree, where each node will have to contain the necessary information to display the corresponding screen. We will also need to create a list of keywords to which we will add the user's answer at each step, as the characteristics described by the dichotomous key will correspond to fields in the SQL plant database. When the user presses an answer button, it will signal which branch of the tree to go down and to add an association of the relevant keyword and the field to which it corresponds to the list. When the user presses the "skip" button, the list of keywords will be passed to the search function to find matches for the characteristics specified so far. Otherwise, we will eventually reach a leaf, at which point the list of keywords as well as the name of the plant "officially" found by the dichotomous key will be passed to the search function, as there will be more plants in the database than just those identified by the dichotomous key, so there may be another plant that shares all of the characteristics that the user has just defined that we will only be able to find by querying the database (this way the user is guaranteed to get the result sanctioned by the verified key as well as other potential matches).

Keyword Search:

The user is given options of fields by which they can search, such as flower color and location. To simplify the implementation of SQL queries in this version, most of the fields will have a set of predetermined answers from which the user can choose, which will be displayed as graphical buttons with the relevant information (swatches for flower color, silhouettes of leaves for leaf shape, etc.). This will eliminate the need to deal with spelling errors or synonyms. We may change which fields are shown in later versions, depending on user feedback

regarding what fields are most useful to search by. Depending on how many fields the keyword search ends up implementing, we may want to have separate screens with different categories of characteristics (such as appearance and habitat) so that the UI doesn't become too cluttered.

To use the keyword search, the user would select the buttons that closest match the characteristics of the plant that s/he is trying to identify and press a "Search" button, which will then take the user to the list of potential results. The user can choose not to select any buttons in any given field and still receive a list of potential results after hitting "Search."

From a technical perspective, the user's selection of buttons will add associations of keywords and the fields to which they correspond to a list of keywords corresponding to fields in the SQL plant database. Pressing "Search" will signal that the list is complete and send it to the:

Search Function

This function will take a list of keywords and the fields to which they correspond and translate it into a basic SQL query. We want to retrieve the name and unique ID of each plant where the given keywords are stored in the corresponding fields (should be easy to convert our list of keywords to the form SELECT name, ID FROM plant_database WHERE field IN keyword {AND more conditions}+). Later versions should be able to deal with partial results, but v1 will only display results for which all indicated attributes are present.

Potential Results:

We will need to convert the table of plant names and IDs returned by the search function to a results page with a list of plant names, each of which acts as a link to its fact sheet.

Plant Database:

The database we will be using is SQLite. SQLite is public domain, so we can use it without paying. Apple uses SQLite for its iPhone applications, so we know it will be compatible and there will be many resources to help us learn to use it.

Our database will consist of two tables: the Plants table and the MyPlants table. Plants will hold the information about each individual plant, and will be used with the search and dichotomous key functions. The MyPlants table will hold plants which users have marked as favorites, and will take its information from the Plants table.

The database will be stored in two different ways, locally and remotely. Locally, on each user's iPhone, the database will be stored in memory. SQLite allows for locally stored databases, so users will have access to the tables even when out of range of a wireless connection. A remote server will hold the most recent update of the Plants table. When users download the app for the first time, this is where the database will originate. The server will also allow periodic updates to user's phones so that their Plants tables have the most up-to-date information. The updates will leave their MyPlants table intact.

Fact Sheets:

The fact sheet for each plant will be composed of the text description and picture stored in the database (technically, the pictures themselves aren't in the plant database, but links to access the image files, which will be stored in a folder on the phone). When a user selects a plant from either the potential results or MyPlants list, we will use the ID to perform an SQL

query to retrieve the description and picture from the database (this will definitely return one result, as there won't be more than one plant with the same ID and if the plant is in either the results or MyPlants, it must be in the database). There will need to be a function to format and display this information. There will also be a button superimposed on the fact sheet to allow the user the opportunity to store the plant in their MyPlants list.

MyPlants:

The user's personal list of plants that they have saved from their searches. The names, ID, descriptions, and links to pictures (none of the other fields should be necessary, as this version will not allow users to perform searches on the MyPlants list or sort by attribute) will be stored in a local database, since a particular user's list should only be accessible from his/her phone. When the user selects the "like" button on a fact sheet, that plant will be added to the MyPlants database. The user will be able to browse through his/her list of plants, go to the fact sheets for any of the plants on the list, and remove any of the plants.

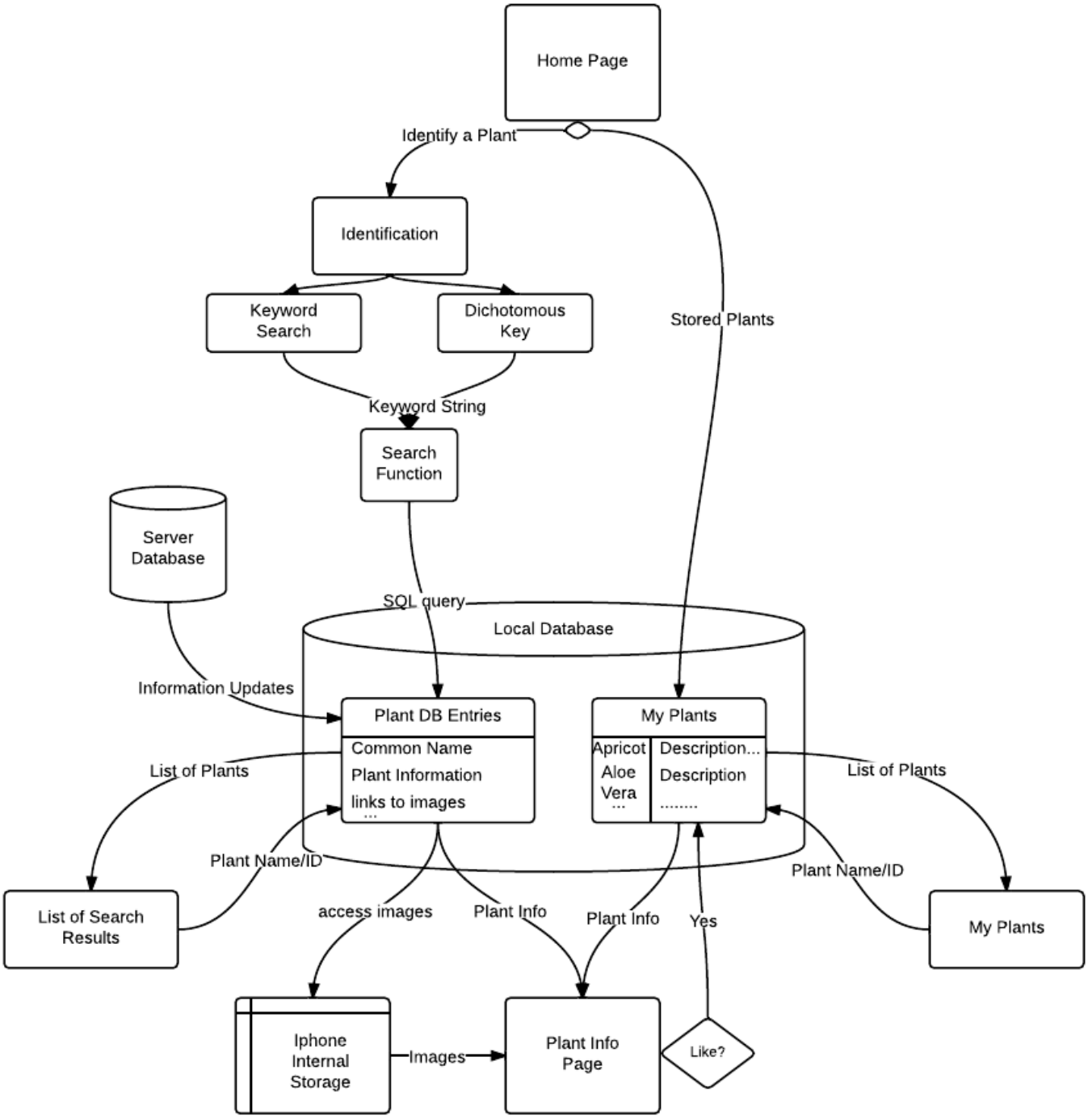
Interfaces

External interface:

Because the iPhone screen is relatively small, we want to make sure that each page has only the necessary features to keep it from becoming cluttered. We are looking at using large buttons and text input boxes to allow for easy navigation throughout the system.

Internal interface:

The primary interface that we are looking at is passing SQL information between internal components. SQLite, the implementation we are using, has good tools to interact directly with iPhone development software – we will be able to pull data out of the databases and format it within our pages without too much trouble as well. As users input text into the keyword search, it can easily be converted to an SQL query using regular database searching tools. The plant information is currently being stored in entries in the database and is unmodifiable by users, but in the future (after version 1.0) we hope to allow users to contribute entries and information as well.


Home Page
Identify a Plant
Stored Plants
Identification
Keyword Search
Dichotomous Key
Keyword String
Search Function
Server Database
SQL query
Local Database
Information Updates
Plant DB Entries
Common Name
Plant Information
links to images
...
My Plants
Apricot
Aloe Vera
...
Description...
Description
.....
List of Plants
List of Plants
Plant Name/ID
Plant Name/ID
List of Search Results
access images
Plant Info
Plant Info
Yes
My Plants
Iphone Internal Storage
Images
Plant Info Page
Like?

Key issues

Initially, we had trouble narrowing down our list of desired features to provide in version 1.0 of our application – we had come up with a very long list of requirements in the planning stage of the project. The features we have decided upon to provide in this version came out of the following rationale: the main functionality of our system was always intended to be plant identification; other features were fun and interesting but weren't necessary to fulfill this basic task.

We initially we had trouble understanding the interactions between the database and the other pieces of the architecture. After our initial plan, the review session and subsequent research has allowed us to have a better understanding of the type of database that is appropriate for the project and the inner-workings of the database structure itself. We also were able to add the specific information that is being passed between different components now that we understand the database better.

Major Issues to Address

At the moment we are using a very basic search algorithm - in the future, we would like to implement a more sophisticated algorithm which would run with more user input (as opposed to predetermined choices for each search field) and be able to return partial matches.

We are currently unsure about how to include pictures in our plant information, because databases cannot hold images. Jesse suggested that we simply put all the images in a file and download it to the phone's static memory in a folder, which may work but seems like it could use up lots of space. We need to balance image size and data size when considering this.

We are still looking into what issues we might have when updating the Plants table from the remote server. Can we be certain that users' MyPlants tables remain intact when the Plants table is updated? This will require looking into database update procedures more thoroughly.