CS221 Group Project

Design Specification

Project Team	Aldridge, William	Atkins, Max
	Barnes, Alexander	Dart, Nicholas
	Harizanov, Maksim	O'Hare, James
	Pocock, Michael	Tuff, Sebastian
	Wilcock, Daniel	Wilmot, Andrew

Version	1.0
Status	Release
Date Published	February 16, 2015
Reference	SE_O2_DS_00

Department	Computer Science
Address	Aberystwyth University
	Penglais Campas
	Ceredigion
	SY23 3DB

Contents

1 Decomposition Description

1.1 Subsystems

The Botanist Tools application is composed of 3 subsystems:

- Android Application
- Website
- Server

1.1.1 Android Application

The android application provides the interface that users will use to record plant data when out on a visit. It implements requirements (FR1), (FR2), (FR3), (FR4), (FR5), (FR6). It must also conform with the requirements (EIR1), (PR1), (PR2), (DC1) and (DC2) [1].

The application will have a form-based activity which can be used to add and edit new and currently saved recordings. Currently saved recordings will be stored locally in a collection, which will be awaiting dispatch to the server. The user will be able to view this list and select recordings to perform actions on (Eg. edit and delete). The application will not show recordings that are saved on the server.

The Android application will communicate with the server to perform functions such as sending recordings and performing user authentication. The application does NOT communicate directly with the website and the database. It uses a web API which is core to the server.

1.1.2 Server

The server will consist of two parts:

- A database
- A web API

1.1.3 Database

The database will be the central datastore for the entire system. It will communicate exclusively with the web API and serve as its back-end.

1.1.4 Web API

The web API is central to the system; it provides a uniform way of accessing the database for all subsystems to use. It maintains the integrity of the datastore by acting as the "middle-man" so that the other subsystems do not damage the contents. It exposes a public interface to allow a set of actions to be performed by the users of the API; actions include user authentication and recordings management. The web API will implement the requirement (FR7) and must also conform to the requirement (PR2).

1.1.5 Website

The website will consist of a set of web-pages which implement all the required functionalities for the user (FR8) and (FR9). It must also conform to the requirements (EIR1), (PR1) and (PR2). The website will communicate with the Web API via HTTP to receive from and send data to the database. The website will have no communication with the Android application. It will also not directly communicate with the database, but will go through the web API.

1.2 Significant Android Components

1.2.1 Significant UI classes

HomeActivity	This class will hold the code to allow a user to move on to the NewRecordingActivity via a startNewRecordingButton.
NewRecordingActivity	This class will hold the code to allow a user to enter information such as Name, Phone Number, E-Mail, and Site Location. It will also allow the application to receive date and time information from the Android device. The nextButton will then move the user to the AddNewSpeciesActivty.
AddNewSpeciesActivity	This class will hold the code to allow a user to enter all the required information about a species entry in the current recording. It allows you to choose the name of a species from a locally saved list, and also allows the user to add a new species name, if not found. This activity must have the functionality to use the device's GPS capabilities to record location information of the species. It allows the user to select abundance in accordance with the DAFOR scale. The user should be able to add a scene/specimen picture through the device's camera or the gallery application. The user can also add a note if they wish. There is then a confirmButton which adds the species to the current recording and moves the user on to the MaintainRecordingActivity.
MaintainRecordingActivity	This class will hold the code to allow a user to maintain the current recording. It will contain the functionality to edit any entered species from the collection in the recording. It will also allow the user to delete the current recording, removing all stored species data. The user will be able to save the recording, sending the current recording to the server at the first opportunity.

1.3 Significant other classes

Specimen	This class will hold the code to define all the information a specimen object can hold. Fields to hold this data will be: speciesName, gpsLocation, abundance, scenePicture, specimenPicture, notes. This class will also provide getter and setter methods for the mentioned fields.
Recording	This class will hold the code to define all the information a recording object will hold. Fields to hold this data will be: usersName, usersPhoneNumber, usersEmail, usersAddress, dateTime. This class will contain getters and setters for the mentioned fields.

1.4 Significant Website Components

Navigation	A set of buttons at the top of each webpage that will allow the user to move around the site. When clicking on the desired button, they will be taken to the designated page.
Homepage	This landing page will provide all the general information a user needs to know about the service. It also contains a search bar which will allow the user to filter through the plant database to find what they are looking for.
View Species Page	This page will display every species in the database for the user to view and select. If the user selects a species, they will be taken to the View Chosen Species Page, which provides more species information. This page will also allow the user to add a new species, taking them to the Add Species Page.
View Chosen Species Page	This page will display all information about the specific species chosen by the user.
Add Species Page	This page will provide all the forms necessary to input all data for a chosen species (name, location, abundance, scene picture, specimen pictures and notes), or allow a user to add a new species that is not listed.
Map Overlay	This component will be a pop-up map that will show the locations of the chosen species. This location will be taken from the database.

1.5 Significant Server Components

Server	A database and a set of Web API commands to allow manipulation.
Database	a MySQL database where we will store all data for our system
Web API	A list of commands that allow the Android and website systems to access and manipulate data in the Database.

- Tables
  - botany_users
  - botany_records
  - botany_specimens
  - botany_reserves
  - botany_resources
- addRecord.php - Adds a record to the database
- addReserve.php - Adds a reserve to the database
- addResource.php - Adds a resource (picture) to the database
- authenticateAdmin.php - Checks an inputted password and tells if it's the same as the stored password
- getRecord.php - Returns a record
- getRecords.php - Returns all records
- getReserve.php - Returns a reserve
- getReserves.php - Returns all reserves
- getResource.php - Returns a resource (picture)
- getSpecimen.php - Returns a specimen
- getSpecimens.php - Returns all specimens
- removeReserve.php - Removes a reserve
- removeSpecimen.php - Removes a specimen
- updateReserve.php - Updates a reserve
- updateSpecimen.php - Updates a specimen

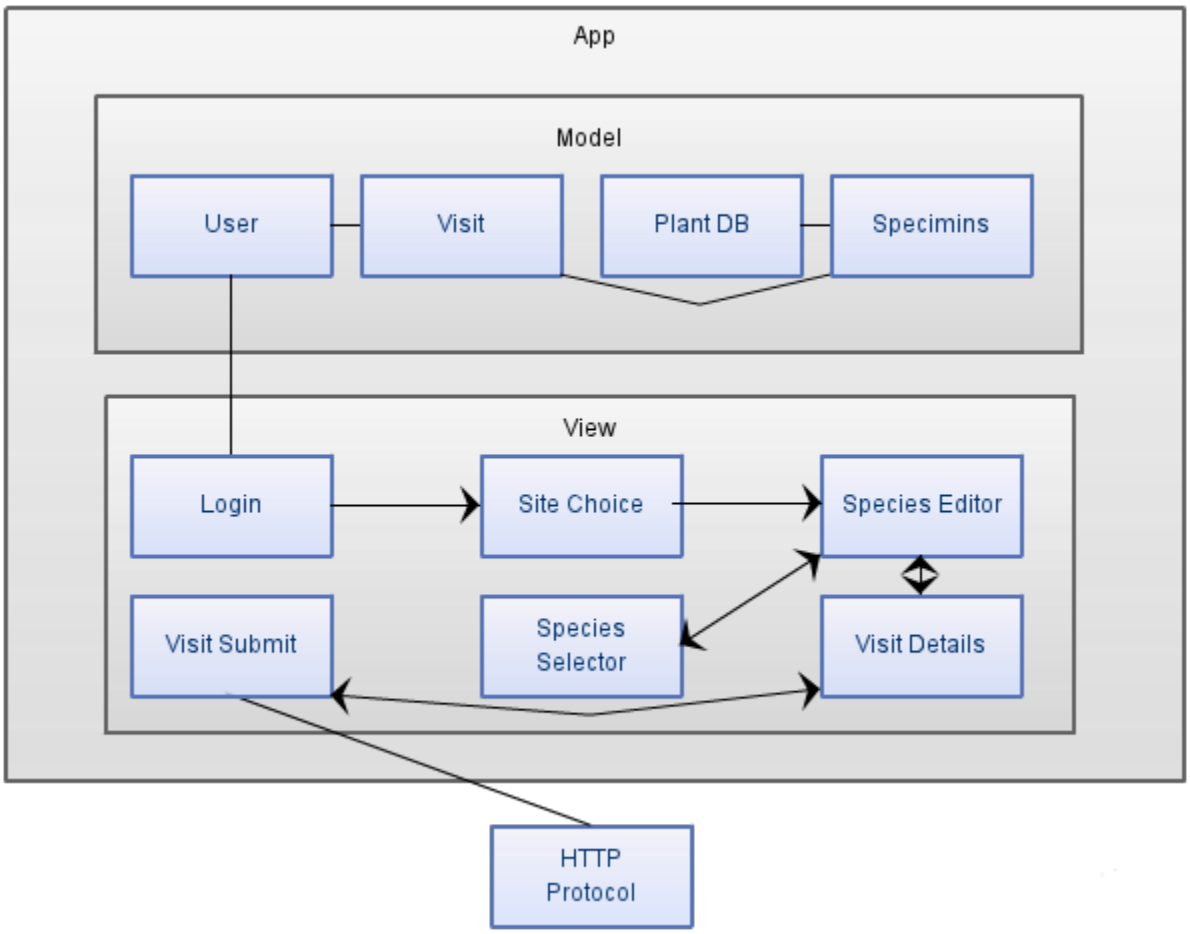

Figure 1: Component diagram for Android

2 Android Application Design

2.1 Decomposition

The compilation dependencies are as follows:

- Android dependencies a user will have an array of visits
- A visit will have an array of specimens
- A specimen can get data from the plant db (Fig 1).

2.2 Interfaces

2.2.1 Login Activity Interface

```
/**
 * A login screen that offers login via email and password.
 */
public interface LoginPage extends Activity implements LoaderCallbacks<Cursor>
    {

    /**
     * Called when building page for the GUI interface;
     */

    protected void onCreate(Bundle savedInstanceState);

    /**
     * AutoFill
     */
    private void populateAutoComplete();

    /**
     * Check email
     */
    private boolean isEmailValid(String email);

    /**
     * Shows the progress UI and hides the login form.
     */
    @TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
    public void showProgress(final boolean show);


    /**
     * Called when building page for the GUI ;
     */
    public Loader<Cursor> onCreateLoader(int i, Bundle bundle);

    /**
     * Called when building page for the GUI is finished;
     */
    public void onLoadFinished(Loader<Cursor> cursorLoader, Cursor cursor);


    public void onLoaderReset(Loader<Cursor> cursorLoader);

    /**
     * Called when building page for the GUI interface;
     */
    private void addEmailsToAutoComplete(List<String> emailAddressCollection);

    /**
     * Represents an asynchronous login/registration task used to authenticate
     * the user.
     */
    public interface UserLoginTask extends AsyncTask<Void, Void, Boolean> {
```

```
        protected Boolean doInBackground(Void... params);

        protected void onPostExecute(final Boolean success);
56
        protected void onCancelled();
    }
}
```

android/src/loginInterface.java

2.2.2 Site Choice Activity Interface

```
public class SiteChoice extends Activity {

    /**
     * Called when building page for the GUI interface;
     */
    protected void onCreate(Bundle savedInstanceState);

    /**
     * Inflate the menu; this adds items to the action bar if it is present.
     */
    public boolean onCreateOptionsMenu(Menu menu);

    /**
     * Handle action bar item clicks here. The action bar will
     * automatically handle clicks on the Home/Up button, so long
     * as you specify a parent activity in AndroidManifest.xml.
     */
    public boolean onOptionsItemSelected(MenuItem item);
}
```

android/src/siteChoiceInterface.java

2.2.3 Species Selector Activity Interface

```
public class SpeciesSelector extends Activity {

    /**
     * Called when building page for the GUI interface;
     */
     protected void onCreate(Bundle savedInstanceState);

     /**
     * Inflate the menu; this adds items to the action bar if it is present.
     */
     public boolean onCreateOptionsMenu(Menu menu);

     /**
      * Handle action bar item clicks here. The action bar will
      * automatically handle clicks on the Home/Up button, so long
      * as you specify a parent activity in AndroidManifest.xml.
      */
     public boolean onOptionsItemSelected(MenuItem item);
}
```

android/src/speciesSelectorInterface.java

2.2.4 Visit Management Activity Interface

```
public class VisitManagement extends Activity {

    /**
     * Called when building page for the GUI interface;
     */
    protected void onCreate(Bundle savedInstanceState);

    /**
     * Inflate the menu; this adds items to the action bar if it is present.
     */
    public boolean onCreateOptionsMenu(Menu menu);

    /**
     * Handle action bar item clicks here. The action bar will
     * automatically handle clicks on the Home/Up button, so long
     * as you specify a parent activity in AndroidManifest.xml.
     */
    public boolean onOptionsItemSelected(MenuItem item);
}
```

android/src/visitManagementInterface.java

2.2.5 Visit Submit Activity Interface

```
public class VisitSubmit extends Activity {

    /**
     * Called when building page for the GUI interface;
     */
    protected void onCreate(Bundle savedInstanceState);

    /**
     * Inflate the menu; this adds items to the action bar if it is present.
     */
    public boolean onCreateOptionsMenu(Menu menu);

    /**
     * Handle action bar item clicks here. The action bar will
     * automatically handle clicks on the Home/Up button, so long
     * as you specify a parent activity in AndroidManifest.xml.
     */
    public boolean onOptionsItemSelected(MenuItem item);
}
```

android/src/visitSubmitInterface.java

2.2.6 User Data Class Interface

```
public class User{

    /**
     * Returns the users first name
     */
    public String getUserForeName();

    /**
     * Returns the users last name
     */
    public String getUserLastName();

    /**
     * Returns the users phone number
     * Formatted accordingly (Eg 07...)
     */
    public String getUserPhoneNumber();

    /**
     * Returns the users email
     */
    public String getUserEmail();
}
```

android/src/userClassInterface.java

2.2.7 Visit Data Class Interface

```
public class Visit{

    /**
     * Returns the visit Date when the user logged in
     * https://docs.oracle.com/javase/7/docs/api/java/sql/Date.html
     * Fulfils requirement FR2
     */
    public Date getVisitDate();


    /**
     * Returns the visit Time when the user logged in
     * https://docs.oracle.com/javase/7/docs/api/java/sql/Time.html
     * Fulfils requirement FR2
     */
    public Time getVisitTime();


    /**
     * Returns a list of all specimens entered so far
     * Filfils FR3
     */
    public Specimen[] getSpecimen();

    /**
     * Adds a specimens to the visit
     * Fulfils FR3
     */
    public void addSpecimen(Specimen s);

}
```

android/src/visitClassInterface.java

2.2.8 Plant DB Interface

This Will be a file obtained from the Botanical Society of Britain and Ireland containing a large list of known plant types in a comma sperated list (CSV) format. This file will be parsed at runtime and a condensed list created in alphabetical order for searching. Obtained from *http : //www.bsbi.org.uk/resources.html*

2.2.9 Specimen Class Interface

```
public class Specimen{

    /**
     * Abundance scale
     * Fulfills FR4
     */
    public Enum AbundanceEnum {
        DOMINANT,
        ABUNDANT,
        FREQUENT,
        OCCASIONAL,
        RARE,
    }

    /**
     * Gets the approximate specimen longitude
     */
    public String getLatitude();

    /**
     * Gets the approximate specimen latitude
     */
    public String getLongitude();

    /**
     * Gets the abundance rating
     */
    public AbundanceEnum getAbundance();

    /**
     * returns the free text comment made by the user
     */
    public String getComment();

    /**
     * returns a URI to the scene photo if one is provided
     */
    public String getScenePhotoURI();

    /**
     * returns a URI to the specimen photo if one is provided
     */
    public String getSpecimenPhotoURI();

    /**
     * takes a specimen photo.
     * Opens up the camera and allows the user to take a photo rather than
```

```
     * asking for the location of one
49   */
    public void takeSpecimenPhoto();

    /**
     * takes a specimen photo
54   * Opens up the camera and allows the user to take a photo rather than
     * asking for the location of one
     */
    public void takeScenePhoto();
}
```

android/src/SpecimenClassInterface.java

3 Web Design

3.1 Interface

3.1.1 config.php

Creates an array that will give the main location to the API and declares the directory that the sessions will be stored in.

3.1.2 header.php

- includes config.php
- Error reporting
  - On the occurrence of an error, error messages are removed and not displayed
- Meta data
  - This declares the meta information for the site such the author and the site description
- title declaration
  - Echos the title of the page that is declared on each of the individual pages
- CSS links
  - Links to all the CSS files
- Login functions
  - If the password session does not exist, then the authentication form is displayed. If the authentication session exists, then a logout is displayed
- Navigation
  - Creates a constant header throughout the website.
- Messages
  - When the authentication or logout method is called, it will either display a success, error, or logout message

3.1.3 footer.php

The footer will be ending specific div tags from the page and the header.

3.1.4 Authenticate.php

- includes config.php
- Check authentication form is submitted
  - The script checks if a password has been submitted and attaches the string to a session.
- Authentication CURL POST request
  - First gets the password session variable and sets the variables.Then uses a variable to connect to the API method through the config file. It will then set the url, the number of POST variables and the data being sent. The server will then reply with a response message stating if the password is correct, it is true, where a message will be generated. If it is false, then a login error message will be sent to the previous page and the session will be destroyed.

3.1.5 index.php

- Title variable
  - The title is attached to a variable that is then called within the header
- Include header.php
- Simple search form
  - This form will search for a species name that matches any of the data that is inputted inside of the form. The user will be taken to the plant species page where the search results will be displayed.
- Include footer.php

3.1.6 reserves.php

- Title variable
  - The title is attached to a variable that is then called within the header
- include header.php
- include reserves_curl.php
  - includes the script that gets all the reserves from the server.
- Declaration of reserve variables

  - A for each loop will go through all of the reserve objects and attach them to a new variable. The new variable will then attach the data for a variable that is easily recognizable and usable.

- Admin functions

  - If the password session exists, then the user will be able to have more options for each reserves. If they are authenticated, then the user can delete and edit the reserve. If they are not, they do not get the option to do this.

3.1.7 reserves_curl.php

- includes config.php

  - includes the config file that holds the information to access the web API and the session directory
  - First it uses a variable to connect to the API method through the config file. It will then set the url, the number of POST variables and the data being sent, which will get the order and method function, and the methods declared for that function. THe data is then sent, and then received back from the server, which will contain the JSON object, which will then be decoded and assigned to a variable for practical use. The request is then sent, and the connection is closed

3.1.8 add_reserve.php

- Title variable

  - The title is attached to a variable that is then called within the header

- Include header.php

- Add reserve form

  - This form will be used to input the data that will be sent to the web API via a POST request.

- addReserve POST request

  - First it uses a variable to connect to the api method through the config file. The page will have a form inside of it that will POST data, that will then be caught within an array. This array will then be encoded into json and this is assigned to a variable. The script will then set the url, the number that is being posted which will be the single record and then get a response. The POST request will then be closed.

- include footer.php

3.1.9 edit_reserve.php

- Title declaration
  - The title is attached to a variable that is then called within the header
- include header.php
- Authentication session confirmation
  - If there is no session, then the user will be redirected to the reserve page.
- include get_reserve.php
- Edit reserve form
  - This form will be the same as the add reserve function, however, the values will be already there from an existing reserve, that will get the reserveID that has been posted. Th reserveID will be in a hidden input type so the ID will be recognized for the editing of the reserve. This form will be used to input the data that will be sent to the web API via a POST request.
- editReserve POST request
  - First the POST request checks that the location name has been submitted, and then it uses a variable to connect to the api method through the config file. The page will have a form inside of it that will POST data, that will then be caught within an array. This array will then be encoded into JSON and this is assigned to a variable. The script will then set the url, the number that is being posted which will be the single record and then get a response. The POST request will then be closed. Once the script has finished, the user will be redirected to the reserves page.
- include footer.php

3.1.10 get_reserve.php

- include config.php
- getReserve POST request
  - First it uses a variable to connect to the API method through the config file. The script will be using a GET method to get the resID that the user is editing. The script will then set the url, the number that is being posted which will be the single record and then get a response. Once the response is available, it will be decoded from JSON to an array to an object. The POST request is then closed.

3.1.11 plant_specimens.php

- Title variable
  - The title is attached to a variable that is then called within the header
- include header.php
- include specimens_curl.php
- include filter.php
- Search form
  - This form allows the user to select a searching filter (species, user name, or location name), and lets them input a textual value for the selected filter.
- Sort form
  - This form allows the user to sort the list of specimens by a selected field and pick sorting order. It also includes pagination controls (next page/previous page).
- Results table
  - The results table is where the results of the current search/sort are being displayed. It is accessed by the client-side script and updated with new rows whenever new search results are ready.
- Client-side scripts
  - Showing the user search results is performed asynchronously, by calling the website back-end using a POST request. The back-end in turn goes to the Web API, and performs a search request. The Web API returns a JSON-formatted array of specimen results, which gets send back to the client-side for processing and visualisation.
  - The script is able to react instantaneously to user input by placing event handlers on controls in both the sort and search forms, triggering a new request whenever a change in values is detected. The search value text field inside the search form makes an exception – instead reacting instantaneously to changes in the text value, the script polls the content of the text field every second, and then decides whether to perform a new request to the back-end, thus avoiding an unnecessarily large amount of requests being sent.

3.1.12 filter.php

- This file starts with a new div tag that will contain the form that will allow the user to filter and sort the results. Once the this div section ends, the pagination will be there, where they will just be buttons.

3.1.13 add_specimens.php

- Title variable
  - The title is attached to a variable that is then called within the header
- include header.php
- include reserve_list.php
- add specimen form
  - This form will allow the user to input the data concerning the specimen. This will then be posted to be available to be used in the POST request.
- include footer.php
- add_resource POST request
  - First it uses a variable to connect to the API method through the config file. It will then set out the options for the CURL POST request. However, if there is no image uploaded, then it will be assigned -1, which will make it that it does not exist. There will be another script basically the same for the other image that will be uploaded.
- Addrecord
  - An array will be created for all of the fields that will be inputted in the form. Before the submit, they will be empty, and once the form is submitted, the form will POST the data into the array. There will be two arrays for this, the first will be specifically for the specimen and the second will be for the user details. The two arrays will be then encoded into json and sent to the API.

3.1.14 reserve_list.php

- This script will use a POST request that will get all of the reserves, and then put them into an array so they can only existing reserves can be selected.

3.1.15 edit_specimens.php

- Title variable
  - The title is attached to a variable that is then called within the header
- include header.php
- include record_curl.php
- Edit specimen form
  - This form will allow the user to input the data concerning the specimen. The existing data for the specimens will already be inputted. There will be a hidden field for the ID so that the specimen that is meant to be edited is updated. This will then be posted to be available to be used in the POST request.

- include footer.php
- editResource POST request
  - First it uses a variable to connect to the API method through the config file. It will then set out the options for the CURL POST request. However, if there is no image uploaded, then it will be assigned -1, which will make it that it does not exist. There will be another script basically the same for the other image that will be uploaded.
- editRecord POST request
  - An array will be created for all of the fields that will be inputted in the form. Before the submit, they will be empty, and once the form is submitted, the form will POST the data into the array. There will be two arrays for this, the first will be specifically for the specimen and the second will be for the user details. The two arrays will be then encoded into json and sent to the API.

3.1.16 specimen.php

- Title variable
  - The title is attached to a variable that is then called within the header
- include record_curl.php
- include header.php
- include img_curl.php
- Specimen checker
  - If a user were to go the specimen page with no id submitted, then they user will be redirected to plant_specimens page.
- LatLong Map
  - Because the map is in a separate page, the lat and the long will be included into a session, where it can then be transferred when the map pop up is shown.
- Specimen table
  - The specimen details will be stored inside of a html table, where they will be presented in a horizontal alignment. Each field data will be received by the record_curl post request include, where it will then be available through through the decoded object. The images will be the same, but with two POST requests for each of the images.
- Default Picture
  - On the occurrence that a specimen does not have an image attached to it, there will be a Jquery script that will give a default image in its place. If there is an error with getting the image, then it will replace the broken image link with the default.

- Session Authentication options
  - If the password session exists, then the user will be able to have two more options for the specimen. They can either edit the specimen or they can delete the specimen. However, if the session does not exist, then the buttons will not exist.
- include footer.php

3.1.17 remove_curl.php

- include config.php
- removeSpecimen POST Request
  - First it uses a variable to connect to the API method through the config file. Then an array will be created that will be storing the id by use of the GET method, and checking if there is a password session.
  - The script will then set the url, the number that is being posted which will be the single record and then get a response. If the response is 200 then the specimen is deleted. The POST request is then closed.

3.1.18 about.php

- Title variable
  - The title is attached to a variable that is then called within the header
- include header.php
- include footer.php

3.1.19 img_curl.php

- include config.php
- getResource POST request
  - There will be two separate POST requests that will do the exact same thing, however, they will be different because there are two pictures that will be recieved. First it uses a variable to connect to the API method through the config file. Then an array will be created that will be storing the resourceID that is declared through the record_curl.php page. The script will then set the url, the number that is being posted which will be the single record and then get a response. If the response is 200 then the specimen is deleted. The POST request is then closed.

3.1.20 logout.php

- include config.php
- Logout function

- As the user is only allowed to have administrative permissions if the password session exists, the user will want to logout. This page basically just destroys the session, which it then creates a message that notifies the user that the user has been logged out.

3.1.21 map.php

- include config.php
- Google Map API
  - The Google map will be in its own separate page because we will be using an iframe to view the map through, that will be viewable on the specimens page. The script first gets the API from Google, then creates the location where it wants to centre onto, which the information will be stored inside of a session. The function is initialised with specific options such as the zoom and the type of map. The map is then set. In the body, the map is then created.

3.1.22 res_remove.php

- include config.php
- removeRemove POST Request
  - First it uses a variable to connect to the API method through the config file. Then an array will be created that will be storing the id by use of the GET method, and checking if there is a password session.
  - The script will then set the url, the number that is being posted which will be the single record and then get a response. If the response is 200 then the specimen is deleted. The POST request is then closed.

3.2 Detailed design

3.2.1 index.php

This is the front page of the website to the user, it contains a description of the purpose of the site.

3.2.2 about.php

This page contains a short description about the app as well as the team.

3.2.3 reserves.php

This page contains a script to display the data from the json object for the reserves. It includes scripts to retrieve the json oject as well as to delete the json ojects.

3.2.4 add_reserve.php

The page gives the user the ability to input data for a new reserve in to a html form. This page also contains a script to collect the data into a json object and a curl method to send the json object to the server.

3.2.5 edit_reserve.php

Edit reserve returns the contents of a json object for reserve to a html form allowing the user to edit its details. A script to encode the reserve to a json object is also included in the page as well as a curl methodto send the json object to the server.

3.2.6 specimen.php

Specimen has a script to return the data from the json object to a table to display as a friendly interface to the user. There is scripts for navigation within the record, these create links to the next and previous specimens within the record. There is a script within the page which produces a pop-up map (using google maps) displaying the location of where this specimen was found. This page also contains scripts to include the site and specimen images.

3.2.7 add_specimen.php

This page contains a html form to collect data from the user about a new specimen to be added. The page also contains a script to make the data into a new json object as well as a curl method to send the data to the server. There is a script to upload images which are then sent to a private folder via a curl method.

3.2.8 edit_specimen.php

Edit specimen contains a script to retrieve the data from the specimen json object and places it in to a html form for the user to edit. There is also scripts to encode the edited data to a json object and to submit the object to the server via a curl method.

3.2.9 plant_specimens.php

Plant specimens contains a script which allows the user to search the database to find a specific species. This page also contains

3.2.10 config.php

The config.php file will be used to establish the connection to the mysql database using the servers API at the start of a new session.

```
$CONFIG=array
api="file location"
session="where the session is stored"
```

The config.php will need to be be included in all php scripts.

3.2.11 authenticate.php

The authenticate.php script will be used to authenticate an admin user on the site. Who through entering a correct password will have access to certain features of the site that regular users will not have. Such as deletion of reserves and specimens. The authentication will be checked to the servers API. If a correct password is entered then the user will see a message confirming it. If an incorrect password is used then a different message will be see saying that a wrong password has been used.

3.2.12 delete_curl.php

This script is used to delete specific specimens from the servers database. This feature will only become available once a user has entered a correct password to show that they are an admin user. PHP CURL is used so that the website can communicate with the server.

3.2.13 edit_specimens.php

This script is similar to the delete script. It uses PHP CURL to communicate with the server. This function is also only available to admin users.

3.2.14 footer.php

This script will be included on every page. It will contain a small website logo in the centre of the footer.

3.2.15 get_reseve.php

This will be used to call all the data about the reserves stored on the server. PHP CURL will be used to communicate with the server and to decode the JSON that the API will be using so that it can be displayed on the reserves page.

3.2.16 header.php

The header will be responsible for creating a session and storing it for the users. The header script will be responsible for declaring the doctype and meta tags such as description, keywords, content type, CSS and font links, validation as well as JAVASCRIPT and the title.

3.2.17 img_curl.php

This is used to find get the specimen and scene photo from the database. PHP CURL is again used to communicate with th server.

3.2.18 logout.php

This script is used to logout an admin user who has entered a correct password to the site. It will end the session that is being used and display a message telling the user that they have been logout of the site.

3.2.19 map.php

The contains the JAVASCRPIT to be able to view the plants location within a Google maps pop up. It will use the latitude and longitude variables of each specimen to give an accurate location.

3.2.20 specimens_curl.php

This will be used to access the information and each individual specimen. CURL is again used to access the server.

3.2.21 resdelete_curl.php

This will work in a very similar way to the delete_curl.php script but will be used for deleting whole reserves instead of just individual specimens. The user will need the be logged in for this function to work.

3.2.22 reserves_curl.php

Much like the record_curl.php it will be used to get information form the server, but information about the reserves rather then the specimen details.

3.2.23 specimens_search.php

This will be used to search for certain specimens within the specimen page on the website. Users will be able to search by Species Name, Location Name and The User Name of people using the site.

3.2.24 filter.php

This will also be used on the plants page on the website and will allow users to order by Species Name, Location, User Name, Date and Abundance and allow results to be show in ascending or descending order.

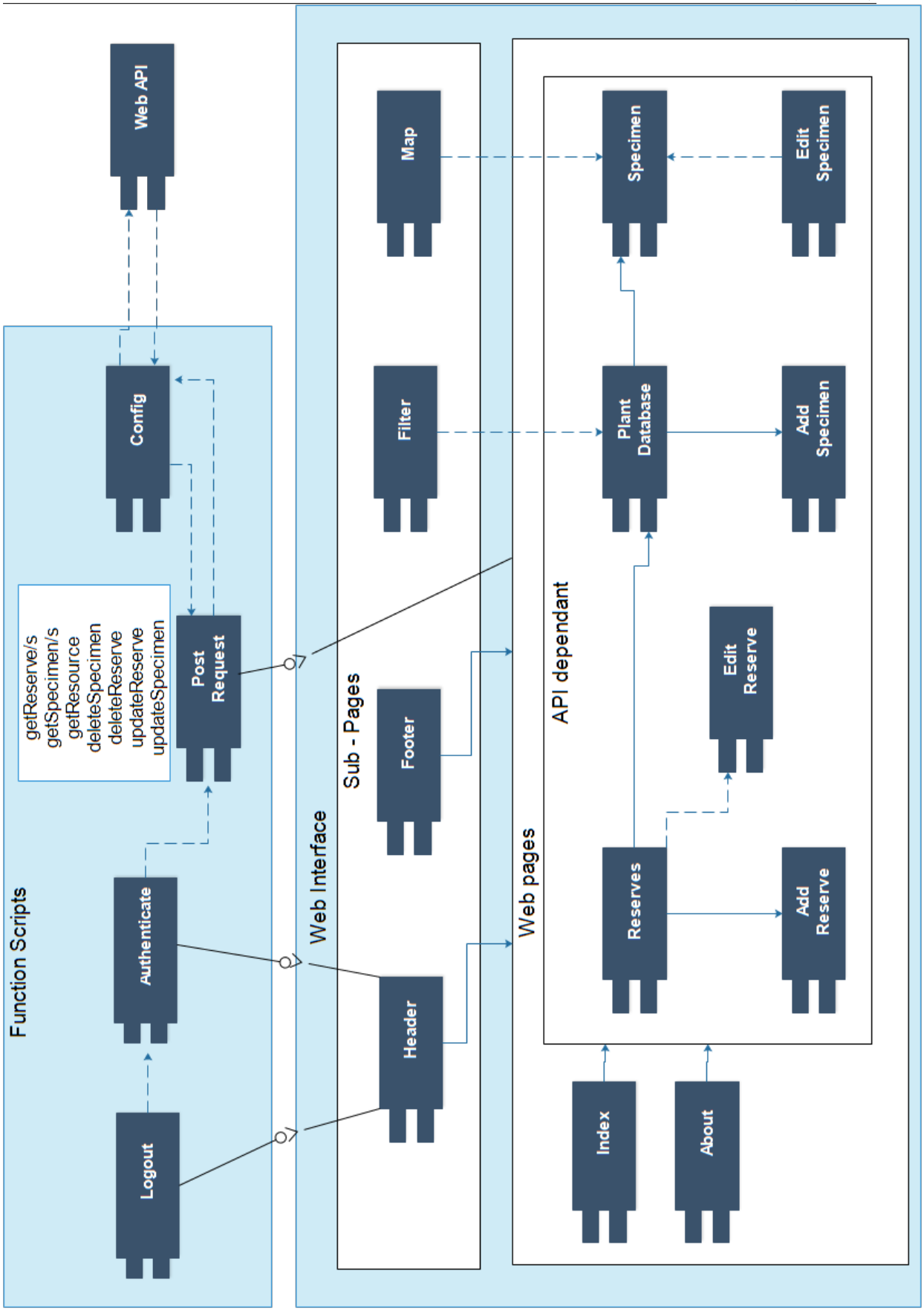

Figure 2: Web component Diagram

4 Server Design

4.1 Interface

These are the commands that the web API will be using to maintain the database and receive commands.

4.1.1 AuthenticateAdmin

Checks the inputted password against the listed valid password to allow website use.

- Headers required:
  - none
- POST arguments:
  - password : String
- Statuses:
  - 200 OK:
    - boolean : true/false
  - 400 Bad Request
  - 500 Internal Server Error

4.1.2 AddRecord

Adds a record to the database, and returns the record ID. Used by the Android application.

- Headers required:
  - none
- POST arguments:
  - record : Record
- Statuses:
  - 200 OK:
    - record ID : Integer
  - 400 Bad Request
  - 500 Internal Server Error

4.1.3 AddReserve

Adds a reserve to the database, and returns the reserve ID.

- Headers required:
  - none
- POST arguments:
  - reserve : Reserve
- Statuses:
  - 200 OK:
    - reserve ID : Integer
  - 400 Bad Request
  - 500 Internal Server Error

4.1.4 RemoveSpecimen

Removes a single specimen from the database, provided an ID and an administrator password. Used by the web site.

- Headers required:
  - none
- POST Arguments:
  - specimenID : Integer
  - password : String
- Statuses:
  - 200 OK : no data
  - 400 Bad Request
  - 401 Unauthorized
  - 500 Internal Server Error

4.1.5 GetSpecimen

Returns a single specimen result, which includes data from the record the specimen belongs to when provided an ID. Used by the web site.

- Headers required:
  - none
- POST Arguments:
  - specimenID : Integer

- Statuses:
  - 200 OK :
    - specimen : SpecimenResult
  - 400 Bad Request
  - 500 Internal Server Error

4.1.6 UpdateSpecimen

Updates a specific Specimen by ID.

- Headers Required:
  - none
- POST Arguments:
  - specimenID : int
  - password : String
- Statuses:
  - 200 OK :
  - 400 Bad Request
  - 500 Internal Server Error

  GetSpecimens Returns a list of specimen results, which include data from the record the specimen belongs to, searching and sorting them by defined criteria. Used by the web site.

- Headers Required:
  - none
- POST Arguments:
  - order : String ("ascending" or "descending")
  - method : String ("speciesName", "locationName", "userName", "timestamp")
  - value : String
  - column : String ("speciesName", "locationName", "userName")
- Statuses:
  - 200 OK : array of SpecimenResult
  - 400 Bad Request
  - 500 Internal Server Error

4.1.7 AddResource

Adds a file resource to the server, and returns an ID. Used by the Android application and the web site.

- Headers required:
  - none
- POST Arguments:
  - resource : OctetStream
- Statuses:
  - 200 OK :
    - resource ID : Integer
  - 400 Bad Request
  - 500 Internal Server Error

4.1.8 GetResource

Returns a file resource when provided with a resource ID. Used by the web site.

- Headers required:
  - none
- POST Arguments:
  - resourceID : Integer
- Statuses:
  - 200 OK :
    - resource data : OctetStream
  - 400 Bad Request
  - 500 Internal Server Error

4.1.9 GetReserve

Returns a single reserve result, which holds data about a location used by records.

- Headers required:
  - none
- POST Arguments:
  - reserveID : Integer
- Statuses:

  - 200 OK :
    - reserve : Reserve
  - 400 Bad Request
  - 500 Internal Server Error

4.1.10 GetReserves

Returns a list of distinct locations used by records in the database for use by the Android application.

- Headers Required:
  - none
- POST Arguments:
  - none
- Statuses:
  - 200 OK : array of Reserve
  - 400 Bad Request
  - 500 Internal Server Error

4.1.11 UpdateReserve

Updates a specific Reserve by ID.

- Headers Required:
  - none
- POST Arguments:
  - reserveID : int
  - password : String
- Statuses:
  - 200 OK :
  - 400 Bad Request
  - 500 Internal Server Error

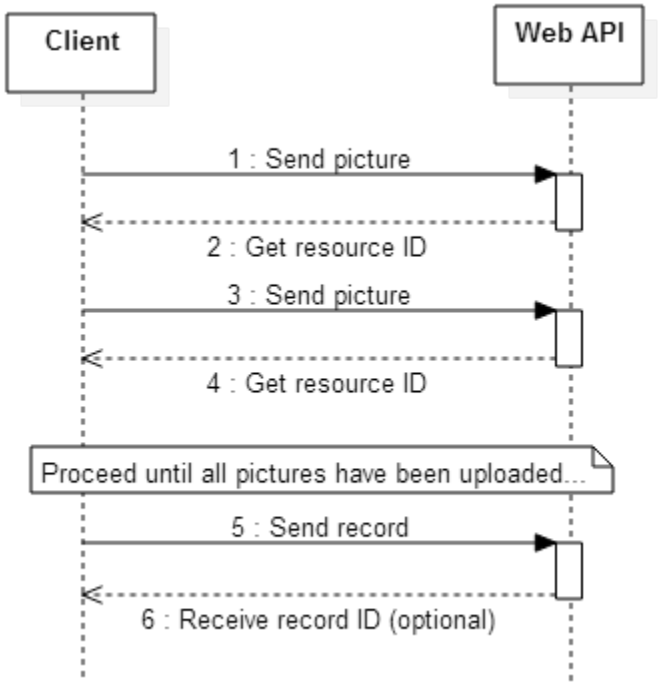

Figure 3: Sequence diagram: adding a record through the web API

4.1.12 RemoveReserve

Removes a specific Reserve by ID.

- Headers Required:
  - none
- POST Arguments:
  - reserveID : int
  - password : String
- Statuses:
  - 200 OK :
  - 400 Bad Request
  - 500 Internal Server Error

4.2 Detailed design

4.2.1 Diagrams

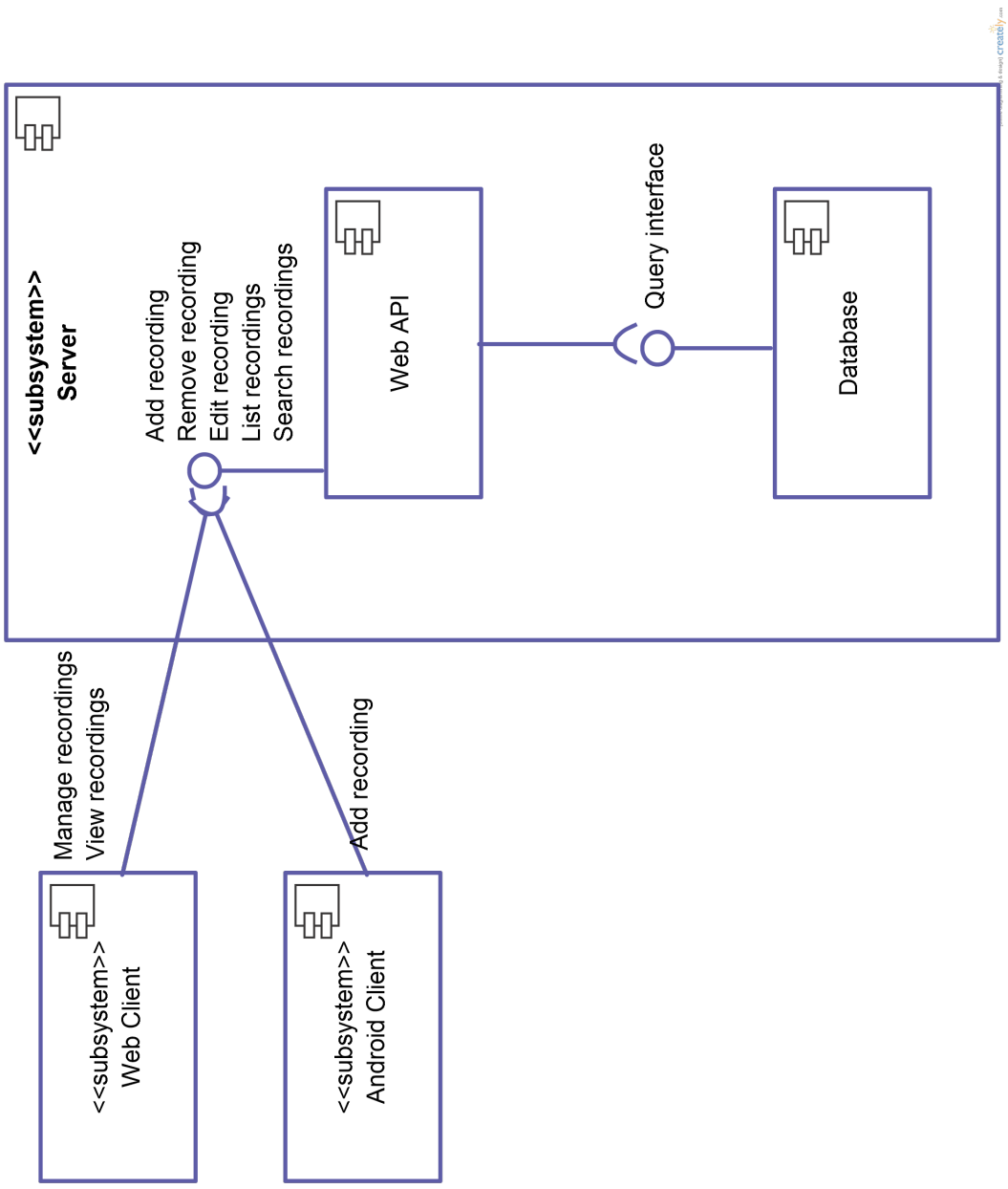

Figure 4: Web Api component Diagram

4.2.2 Significant data structures

The web API is going to make use of the Record, Specimen, RecordLocation and SpecimenResult data structures, which will be exchanged between the server and its clients (the Android application and the website). They will be represented in a JSON format, which is readily available for use in PHP, JavaScript, and Android. Data types, where specified, are JSON data types.

The structure of a Record is as follows:

- Record : Object
  - UserName : String
  - UserPhone : String
  - UserEmail : String
  - LocationName : String
  - Timestamp : Number
  - LocationOS : String
  - Specimens : Array of Specimen

The structure of a Reserve is as follows:

- Reserve : Object
  - ReserveID : Number
  - LocationName : String
  - LocationOS : String
  - Description : String

The structure of a Specimen is as follows:

- Specimen : Object
  - SpeciesName : String
  - LocationLatitude : Number
  - LocationLongtitude : Number
  - Abundance : Number
  - Comment : String
  - ScenePhoto : String (ID of a resource on the server)
  - SpecimenPhoto : String (ID of a resource on the server)

The structure of a SpecimenResult is as follows:

- SpecimenResult : Object
  - UserID : Number
  - RecordID : Number
  - SpecimenID : Number

  - UserName : String
  - UserEmail : String
  - UserPhone : String
  - LocationName : String
  - LocationOS : String
  - Timestamp : Number
  - SpeciesName : String
  - LocationLatitude : Number
  - LocationLongtitude : Number
  - Abundance : Number
  - Comment : String
  - ScenePhoto : String (ID of a resource on the server)
  - SpecimenPhoto : String (ID of a resource on the server)

The web API uses config.php as a configuration store. It contains the $CONFIG array, containing the following entries:

- dbname : the address of the MySQL database to be used
- username : the user name to use on the MySQL database
- password : the password to use on the MySQL database
- adminPassword : the password used in the DeleteSpecimen web method

The web API is going to use a relational database as a data store. The database tables are as follows:

- Users
  - UserId : INT auto-increment PK
  - UserName : VARCHAR(20) not-null
  - UserFullName : VARCHAR(50)
  - UserPhone : VARCHAR(20)
  - UserEmail : VARCHAR(50)
  - UserPassword : BINARY(20)
- Records
  - RecordId : INT auto-increment PK
  - UserId : INT not-null
  - LocationName : VARCHAR(50)
  - Timestamp : INT
  - LocationOS : VARCHAR(10)

- Resources
  - ResourceId : INT auto-increment PK
- Reserves
  - ReserveID : INT auto-increment PK
  - LocationName : VARCHAR(50)
  - LocationOS : VARCHAR(10)
  - Description : TEXT
- Specimens
  - SpecimenId : INT auto-increment PK
  - RecordId : INT not-null
  - SpeciesName : VARCHAR(255) not-null
  - Latitude : FLOAT(10,6)
  - Longitude : FLOAT(10,6)
  - Abundance : INT
  - Comment : TEXT
  - ScenePhoto : INT
  - SpecimenPhoto : INT

References

[1] Project Plan Specification Standards :B. P. Tiddeman (2014-09-23) SE.QA.05b 1.2

5 Document History

Version	Edit	Date	Persons
0.1	Initial Version	November 5 2014	nid21
0.2	Updated with decomposition description	November 12 2014	nid21
0.3	Updated with Android interfaces	November 17 2014	nid21
0.4	Included web and server sections	November 27 2014	nid21
0.5	Document review	November 28 2014	nid21
0.6	More document reviewing and preparing for release	December 02 2014	nid21
0.7	Updated web design spec	Febuary 11 2015	jao14
0.8	Updated web interfaces	Febyary 15 2015	nid21
1.0	Release	Febuary 16 2015	nid21