v1.0

30/10/2014

Design Document

CS22120 GROUP PROJECT

GROUP 3

TOP KEK | LLANDINUM BLDG, PENGLAIS, ABERYSTWYTH, DYFED SY23 3DB

Authors

Cormac Brady

Zach Yewman

James Portch

Scott Lockett

Contents

1 Introduction

1.1 Purpose of this document

This document describes the outline design for the Software Engineering Group Project 2014 - 2015. It should be read taking into account the details of the group project assignment and the group project quality assurance (QA) plan [1].

1.2 Scope

The design specification breaks down the project into separate components and describes the interfaces between those components. It is created according to Requirements Specification SE.QA.RS 2014 -2015 for the group project. References to specific requirements are given in round brackets, e.g. (EIR1). This document is to be read by all members of the team involved in implementation.

1.3 Objectives

The objectives of this document are to:

- Describe the main components of the Reserve Plant Species Recording
- To show the dependencies between components
- To provide details for how each of the main components interface

2 ARCHITECTURAL DESCRIPTION

2.1 Programs in system

2.1.1 RPSRrec (Android program)

The Android Application provides the interface to enable users to make records on the different plant species found in a nature reserve, and then submit that record to a database. This program implements the requirements (FR1), (FR2), (FR3), (FR4), (FR5), and (FR6).

The program will be able to collect multiple plant records and then send the records to the database where it can then be stored, and viewed from the website. It must also be able to use the phones GPS, picture gallery, and camera as methods of getting the users location, and also getting pictures to put into each plant record.

2.1.2 RPSRview (Web Interface)

The web interface allows users to create, view and maintain existing records, it also allows users to maintain reserve details and update as necessary. The website ensures that users can access the database and view not only their own recordings but all recordings of plant species for reserves. This program implements the requirements (FR8) and (FR9).

The website will provide for creation, update and deletion of reserve records and then send the records to the database where it can then be stored. The website will enable users to select a reserve to view a complete list of species recorded at that site including the abundance of the species in question along with any photographs and information on the reserve the record was collected from.

2.1.3 Server (RPSRsrv)

The server will receive recordings from the Android Application and add the recordings to the database. The server will receive the recordings through HTTP posts of MIME messages; these will contain the information about the visit and the information about the species recorded. This program implements the requirement (FR6) and (FR7).

The server will add all recordings to the database which can then be displayed and altered from the website, the server will keep track of all alterations and deletions from the app and the website and maintain the database in accordance. All HTTP posts to the server will be sent to a predefined URL which will be integrated into both the app and the website.

2.2 Significant classes

2.2.1 RPSRrec (Android program)

AddRecord: This will be the class used for adding new records.

SubmitRecord: This will be the method that communicates with the database and sends the records to it.

UserInfo: This will be class that collects the user's information.

2.2.3 Server (RPSRsrv)

ClientUpdates: This will be used to serve any RPSRrec function required from the server

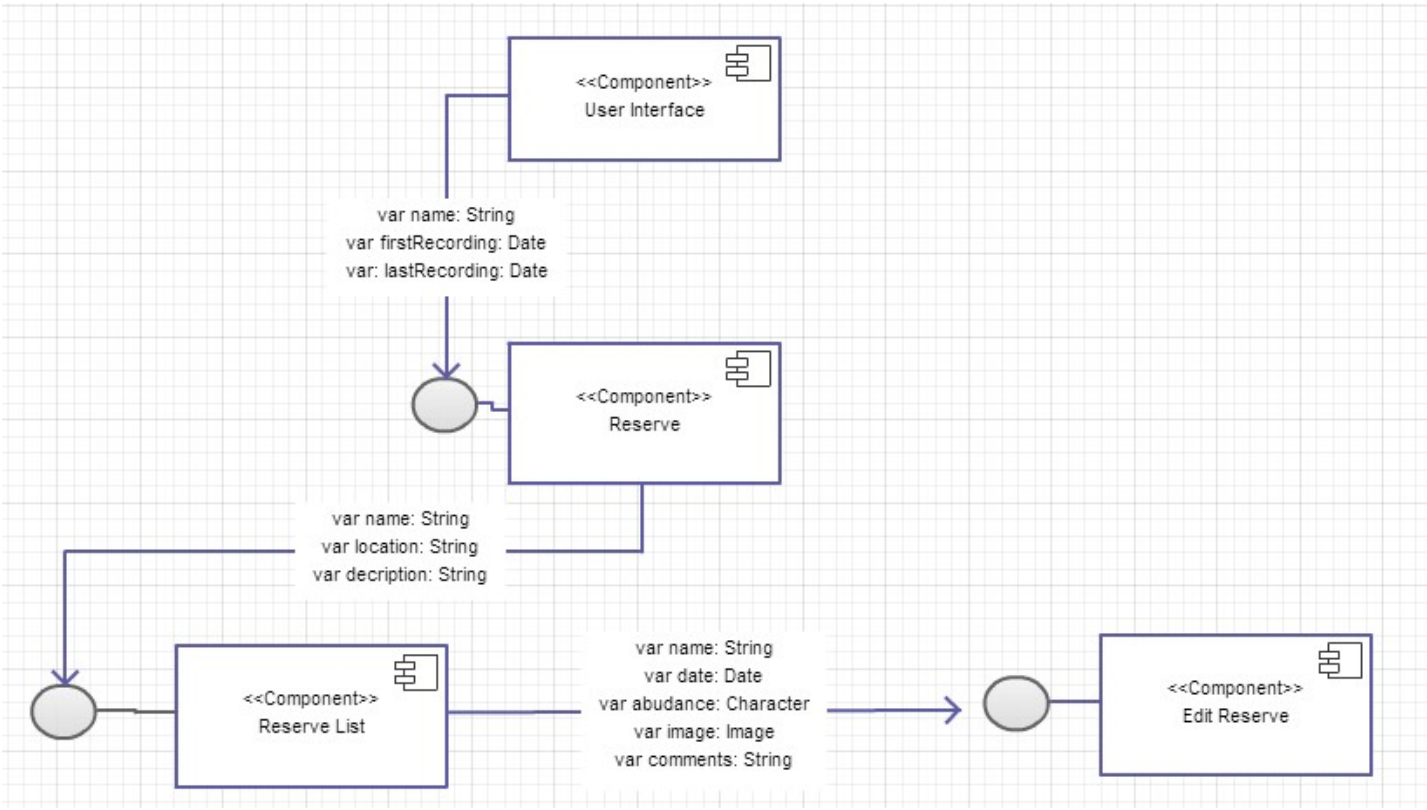
AutoUpdates: changes any information (such as species) that can be done without user input

webUpdates: This will be used to serve any RPSRview functions required from the server

3 Dependency description

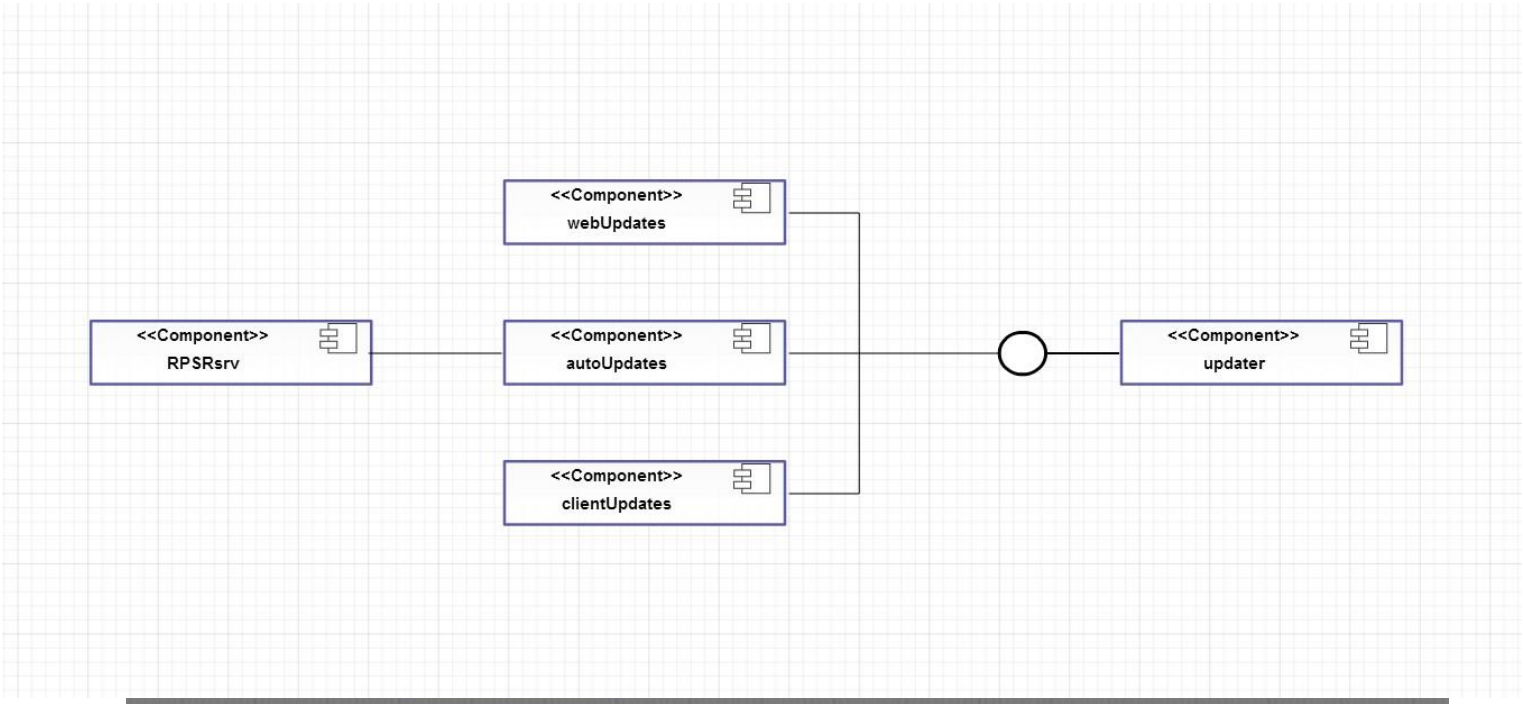
3.1 Component Diagrams

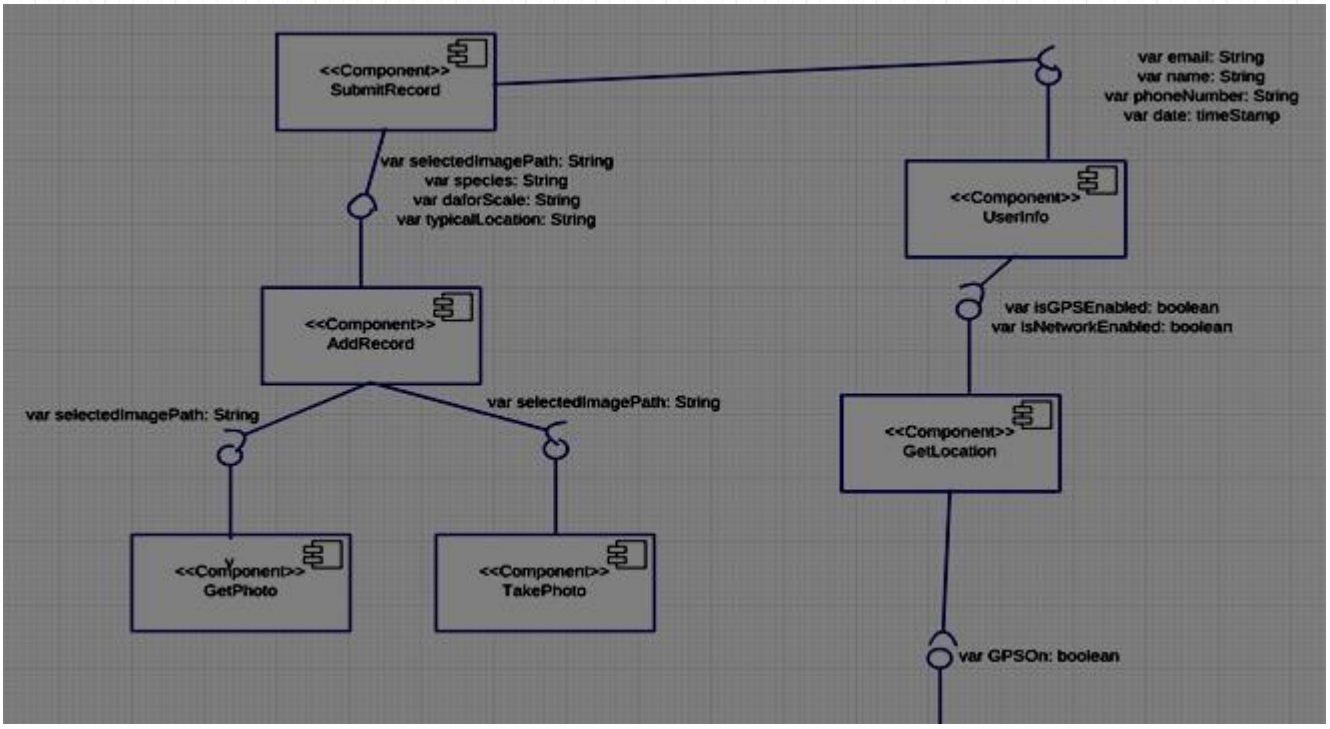
3.1.1 Components Diagram for Web Interface (RPSRview)



3.1.2 Components Diagram for Server (RPSRsrv)

3.1.3 Components Diagram for Android application (RPSRrec)





3.2 Compilation / inheritance dependencies

3.2.1 For Android application (RPSRrec)

“SubmitRecord” depends on “UserInfo” and ”AddRecord”.

3.2.2 For Server (RPSRsrv)

Auto updates is dependent on RPSRsrv, all class dependant on their interface

3.2.3 For Web Interface (RPSRview)

Reserve is dependent on Reserve List and Edit Reserve on Reserve. Are all dependent on the server class webUpdates().

4 Class interface description

4. 1 Android application (RPSRrec)

```
Public class UserInfo {

    private String email, name, phoneNumber;
private final Date timeStamp;     private
final Location location;

    /**
*     Constructor for the user's information with all the tributes  ini-
tialised
*     @param n name of the user
*     @param e email of the us
*     @param pm phone number of the user
    */
    public UserInfo( String n, String e, String pn);



    /**
*     The following getters and setters methods will be used to set varia-
bles to the
*     users inputs for the fields. The getters methods will then be called
when sending
*     off the recording session to be used as a label for it
*/
    public void setName(String userName);

    public String getName();

    public void setEmail(String userEmail);

    public String getEmail();

    public void setPhoneNumber(String userEmail);

    public String getPhoneNumber();

    /**
```

```
* This method will take a parameter for whether or not the phone has a
GPS signal
* and then depending on the answer will get the users location either by
  their input    * or through the GPS
    */
    public void GPSConnection(boolean GPSon);

    /**
* This method will return a string containing all of the variables set in
  this class          *aswell as the location of the user. It will then be
  called from class sending to the    * database to add the information to
  the records.     */
    public String toString();

}
```

This class will be used primarily for the app page that asks for the user's information. When the user inputs them the setter's methods will set the corresponding variable to the user input which can then be accessed through the getter methods from other classes.

```
public class AddRecord {

    private String species, daforScale, typicalLocation, additionalInfo;
    /**
*     The constructor for this class
    */
    public addRecord(String s, String ds, String tl, String ai);
    /**
*     Getter and setter methods for all the variables the user gives from
the
*     add record page on the app
    */
    public void setSpecies(String speciesName);

    public void getSpecies();

    public void setdaforScale(String speciesName);

    public void getDaforScale();

    public void setTypicalLocation(String speciesName);

    public void getTypicalLocation();

    public void setAdditionalInfo(String speciesName);

    public void getAdditionlInfo();

    /**
*     Get the desired photo of the species and add it to the record
    */
    public void getSpeciesPhoto(File pic);

    /**
*     Get desired photo of location and add it to the record
    */
    public void getLocationPhoto(File pic);
```

```
    /**
```

```
    *This method will return a string containing all of the variables in
this class to be
* used when sending the record off to the database.
    */
    public String toString();
}
```

This class will primarily be used for the app page where the user inputs their record for a species. It will use other classes to get the pictures needed.

```
public class SubmitRecord {    private String
    /**
* This method sends the completed record to the database
* The record will be sent as a Json
    */
    public void sendtoDatabase(ArrayList record);

    /**
* This method will try and connect to the database and if it gets a re-sponse
  will
* return true and will start the sending method
    */
    public boolean connectToDatabase();

}
```

This class will be used to communicate with the server and to send to the database the records. The method “connectToDatabase()” may be better served as a void instead but for now I left it as a Boolean.

```
public class GetLocation implements LocationListener {
    //GPS status
    public boolean isGPSEnabled = false;

    //network status
    public boolean isNetworkEnabled = false;

    //final check for GPS
    public boolean canGetLocation = false;



    //Attempts to get the users location
public Location getLocation();

    //Stop using the GPS
public void stopUsingGPS();

    //function to get latitude
public double getLatitude();

    //function to get longitude
public double longitude

    public boolean canGetLocation();

}
```

This class will use the phone GPS to return the coordinates of the user so that they don't have to input it themselves.

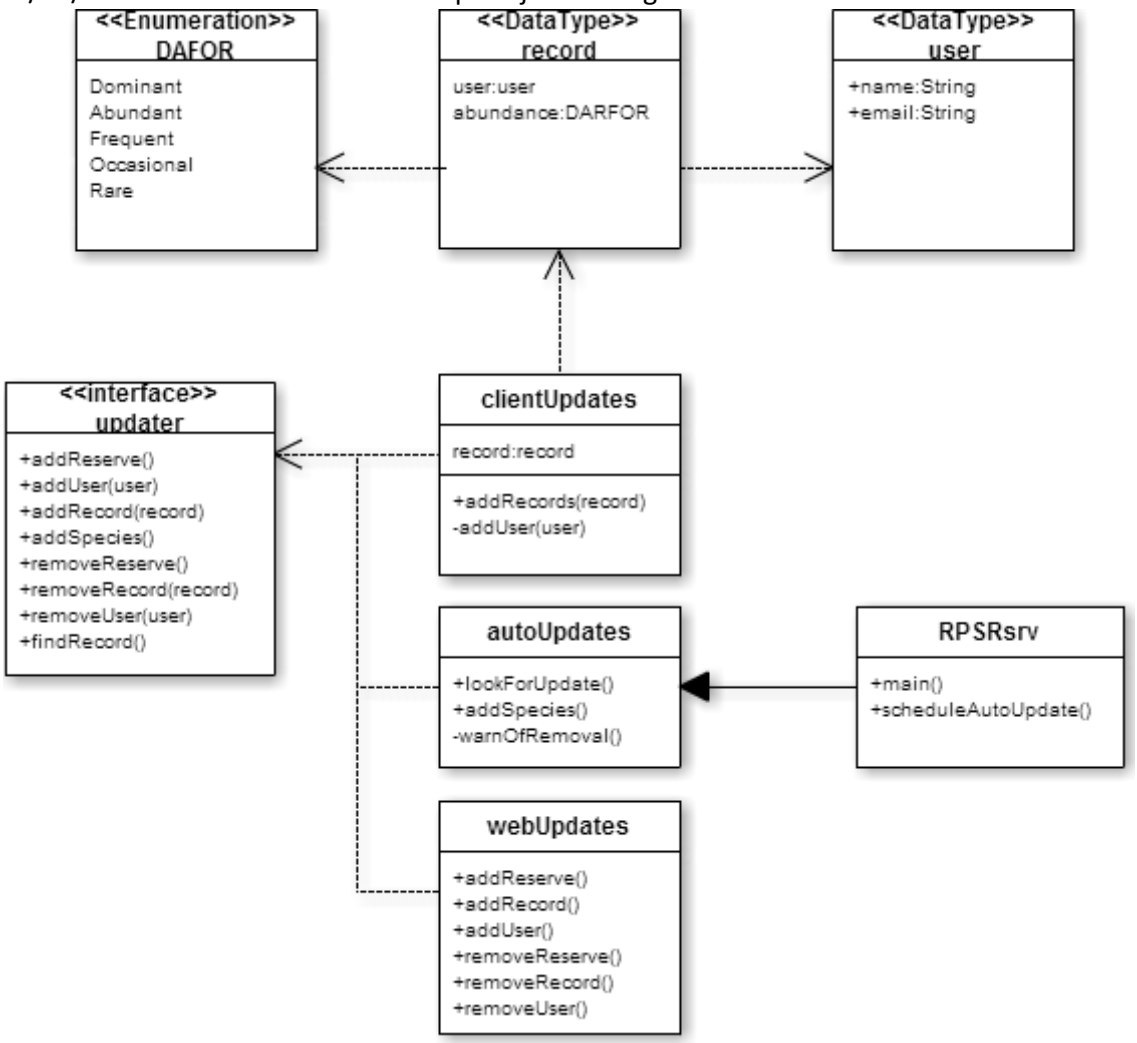
```
public class GetPhoto extends Activity {

    private String selectedImagePath;

    //Choose the picture from the gallery
    public void onCreate(Bundle savedInstanceState);

    //gets the path for the image
    public void onActivityResult(int requestCode, int resultCode, Intent
data);

    //helps retrieve the path of the image URI
public String getPath(Uri uri);
}
```

This class will be used to access the gallery so that the users can select a picture for the record.

```
public class TakePhoto extends Activity {

    private String selectedImagePath;

    //Called when activity started, takes pictures and stores them     public
void onCreate(Bundle savedInstanceState);

    //gets the path for the image
    public void onActivityResult(int requestCode, int resultCode, Intent
data);

}
```

This is the class that will be used to get a picture straight from the camera and not one that can already be found in the gallery.

4. 2 For Server (RPSRsrv)



UML Method Descriptions

RPSRsrv

Main()	
scheduleAutoUpdate()	Schedules an update to happen automatically

autoUpdate

lookForUpdate()	System looks for any changes that may have occurred in order to make sure everything is updated
addSpecies()	Automatically adds a species to the Species table of the database
warnOfRemoval()	Warning message of data removal / over ride

webUpdates

addReserve()	Adds new reserve to the list of reserves
addUser()	Adds a new user to the Users table of the database
removeReserve()	Removes a chosen reserve from the list of reserves
removeRecord()	Removes the all the data for a specific data entry in the Recordings table
removeUser()	Removes a user from the User table of the database
findRecord()	Looks for a specific record in the Recordings table of the database

clientUpdates

addRecords()	Add records to the table of Records in the database
addUser()	Adds a user to the table of Users in the database

5 DETAILED DESIGN

5.1 Android application (RPSRrec)

The RSPRrec program is used to record plant records and then send them to the database, so that they can then be picked up and displayed by the website. Some of the difficult parts of this design for the android app include: connecting to, and sending the plant records and user information to the database, and also using the android api to get images from the users phone gallery or taking a new picture using the phone camera, the app will also need to use the api to get the users location from the phone gps.

Connecting to database: Connecting to the database and then sending the records to it is all done in the SubmitRecord() class. Within this class are two methods, one that connects to the database, and the other that uses the connection to send the records to the database. The most difficult part of this will be initial connection to the database.

Using image from Gallery: Getting an image from the phones gallery is done using the getPhoto() class. Where the gallery needs to be brought up, and then allow the user to select their desired image. Once they have chosen their desired image the path to that image will then be saved. The hard part of this will be getting loading up the gallery and then getting the path to the selected image.

Using image from camera: Taking a picture with the camera, and then getting the images path is done in the TakePhoto() class. This needs to bring up the camera, and then allow the user to

take a picture to be used in the plant record. Much like with getting an image from the gallery, the hard part of this function will be to call the camera from the android api and then get the path of the picture that the user takes.

Using GPS: Using the GPS in the program is done using the GetLocation() class. The class checks to see if the user has their gps, and network enabled. If they are not enabled then this will be passed to the UserInfo() class where it is used so that a drop down list of locations will come up instead of the GPS location of the user. Within the class is a method by the name of getLocation() that needs to get the users location from the GPS which could prove to be quite difficult.

5.2 RPSRview (Web Interface)

The web program will be written in HTML with a CSS file for presentation. The main page will reserve page will contain a HTML table with each of the elements of the reserves within. The edit reserve page will be in same format of a table containing five major additional constituents. The tick boxes at the side of the record which allow the user to select individual records. The delete selected button which will in turn remove the record/records which are currently selected with the tick boxes. A Cancel button which returns the user back to the previous page, without saving any change (if any) that have been made. An apply button which when clicked updates the database and saves the new data which has been input. Finally, textboxes which contain the already present data and will allow the user to edit any of the already input data.

The plant list page will have a heading of the reserve that the user is currently looking at. Again it will contain a table containing the plants name, first recording and its last recording. The name of the plant will be hyperlinked to that particular plant on the recorder list page. This page will contain two buttons. One being the Add entry button which links the users to the add entry page and the edit list button which links the user to edit reverse page.

The recorder list page will have a header being the reserve list with the name of plant selected. This will contain a table which displays the Name of the person who documented the plant along with the date, it's abundance within the reverse, and hyperlink to the image of the plant where it was taken, and any comments about the plant the user made. I will need two button again, one to add a new entry, and another for editing the list, which links to the edit record page, same as the plant list page in terms of functionality.

5.3 Server (RPSRsrv)

The RPSRsrv program is used to receive plant records and enter them into database, so that they can serve to the website (and in a limited scope RPSRrec). A particularly difficult part of this program is the and receiving of the GET POST messages sent from RPSRrec.

Receiving new records: This is done in the clientUpdates() class. Within this class are two main methods, one that connects to the database, and the other that uses the connection to receive the GET POST to process and pass into the other method. The most difficult part of this will be initial receiving.

Updating species list: This will routinely get the spreadsheet file from the required website, check if there is a change, if there is it will proceed to add it into the database this will be done from the auto Updates class but scheduling will be done from the main class RPSsrv.

Viewing/ editing records and adding reserves: viewing the database from a webpage will be handled from the "webUpdates()" class, the website will invoke methods such as "addReserve()" placing input parameters from fields provided to the user on the page.