Group Project Reserve Plant Species

December 2

2014

Design Specification

Group 10

Contents

Copyright Aberystwyth University / Computer Science

1 Introduction

The purpose of this document is to help with the final design of the project and to keep track of the algorithms used for the completion of the project. Pseudo code has been provided for the general overview of the project.

1.1 Scope

This document specifies the way java and web algorithms should be handled .Pseudo code is used for this purpose.

1.2 Objectives

The aim of this document is to get a starting point on the way the project should look both on the web side as well as on the java side. Also it will help towards the final design of the project making it easier to do by having the idea of how it should work beforehand.

2 Decomposition Description

Decomposition Description refers to the way the software system is divided in programs and the modules that make it up. It also shows the purpose and function of each program and significant modules.

Copyright Aberystwyth University / Computer Science

Web Significant Files / Pages

The following is a list of all the significant files which will be used for the website.

./home/index.php - Will give an overview of the website including what it's for and how to use it.
./records/index.php - Will display all of the information about the recordings stored in the database.
./reserves/index.php - Will display all of the information about each of the reserves stored in the database.
./login/index.php - Will display a login form for authorized users.
./include/header.php - The pages header including navigation links and the company logo.
./include/footer.php - The pages footer including information about the company.
./include/php/fetch_data.php - This file will receive recordings via HTTP POST sent by android devices and will validate then store the recordings in the database.
./include/php/session.php - This file will be included by every page and is responsible for maintaining the users session.
./include/php/connect.php - This file will be included by every page that requires a database connection, its purpose is to set up a connection to the mysql database.
./include/php/functions.php - This file will contain useful php functions which are likely to be used a lot by other pages.
./include/css/style.css - The main CSS stylesheet for the website, responsible for design and appearance.

Requirement File providing requirement

FR7 fetch_data
FR8 reserves/index
FR9 records/index

Data Structures

Database Tables

RESERVES

rsv_id | rsv_name | rsv_loc | rsv_desc

The reserves database table will contain information about the different nature reserves that are available for the android application users

to choose. Each reserve record has a unique id, a name, a location and a short description describing the reserve.

RECORDINGS

rec_id | rec_sess_id | users_name | users_phone | users_email | time | species | location |

Copyright Aberystwyth University / Computer Science

abundance | add_info | scene_photo | specimen_photo

The recordings database table will contain the recordings submitted by the android application users. These recordings contain lots

of information about the location, the species being recorded and the person that recorded the information. There are also two columns

(scene_photo and specimen photo) which will store the file locations to the images taken.

SPECIES

species_id | species_name | authority | common_name

The species database table will contain a list of all the species that are available for the user of the android application to select.

This will be populated by the list of plant species provided to us.

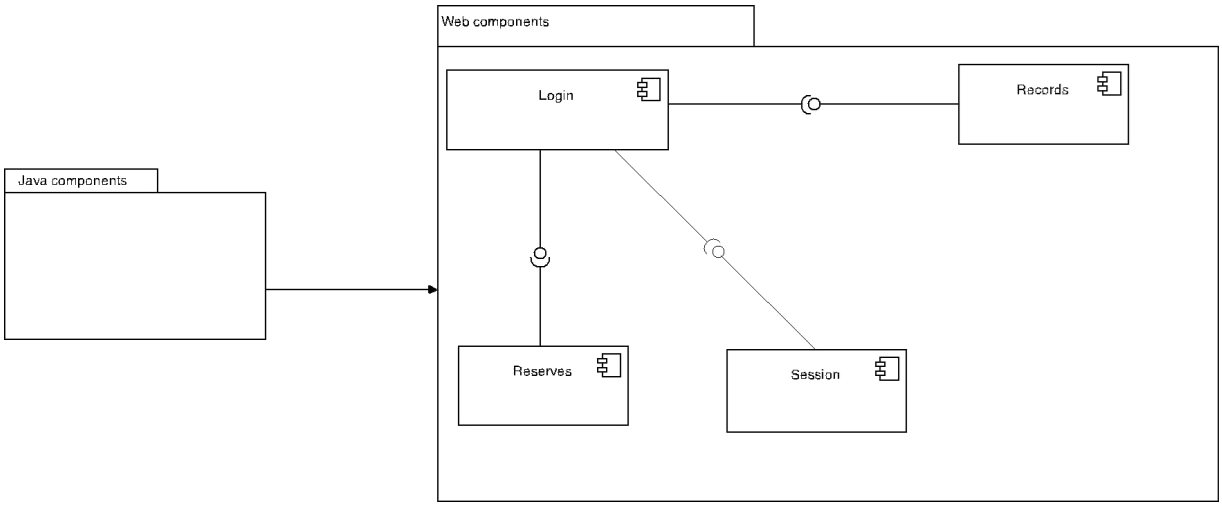
Copyright Aberystwyth University / Computer Science

3 Dependency Description

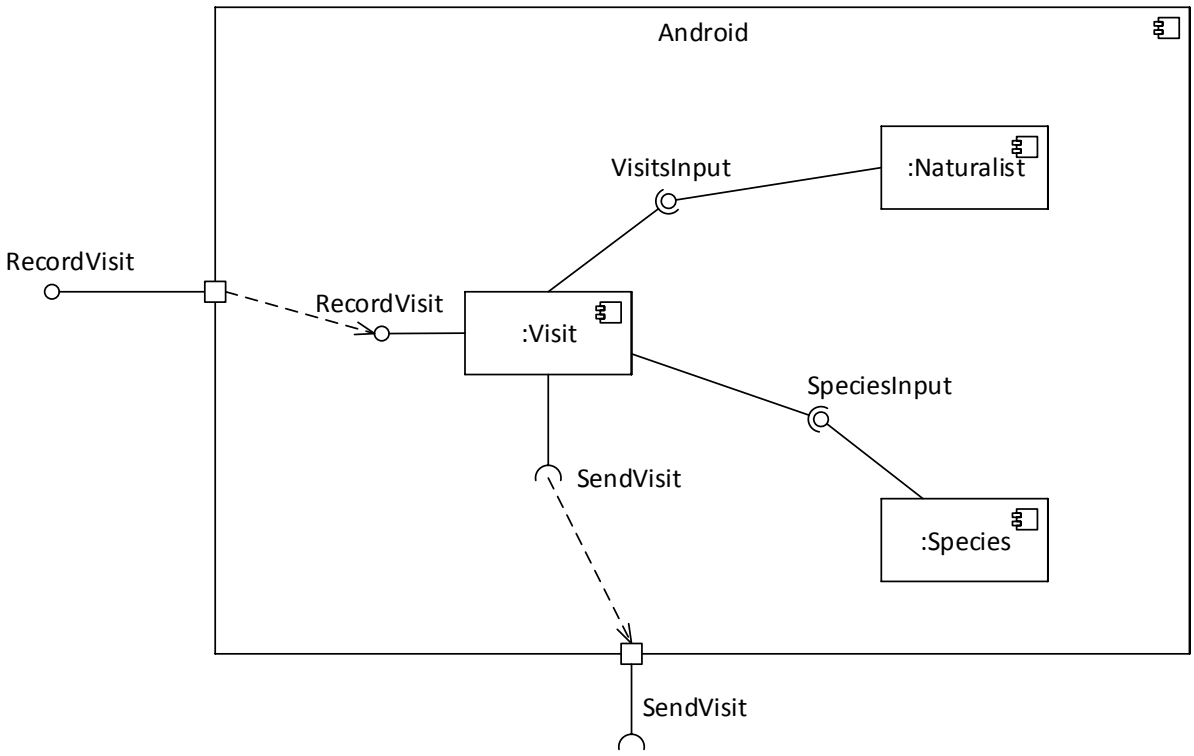
The dependency Description refers to the relationships between modules. It helps with understanding how the system fits together thus describing the general architecture. Component diagrams are a good method of describing the architecture specified above.

3.1 Component diagrams

3.1.1 Component Diagram for Web



3.1.2 Component Diagram for Android app



Copyright Aberystwyth University / Computer Science

3.2 Compilation and Inheritance relationships

There is only one compilation dependency in the Android program:

Visit depends on Naturalist and Species.

4 Interface Description

The interface description should include an outline specification for each class or interface in the system. It is used for informing the designer, programmers and testers everything they need to know in order to use the facilities provided by a module.

4.1 Plant Catalogue Interface Description

```
package uk.ac.aber.cs22120.group10.plantcatalog;

/** Interface for the main class.

 *  Contains methods needed when the application boots up.

 *  Creates instances of naturalist, reserves and species data stored.

 *  If local data is available parse these using the obtain methods.

 *  If network connection is true then call the update method.

 */

public interface PlantCatalog {


        /** Some private attributes:

         *

         *  private Naturalist naturalist;

         *  private ReservesData reservesData;

         *  private SpeciesNames speciesNames;

         *  private SendData sendVisit;

         *  private Visit visit;

         */


        /** Obtains local reserves data.

         *
```


Copyright Aberystwyth University / Computer Science

```
 *  Creates an instance of ReservesData,
 *  if local XML file is available uses method
 *  inside this instance to parse the content.
 *  @return reservesData new instance of ReservesData created.
 */
public ReservesData obtainReserves();

/** Obtains local species names.
 *
 *  Creates an instance of SpeciesNames,
 *  if local XML file is available uses method
 *  inside this instance to parse the content.
 *  @return speciesNames new instance of SpeciesNames created.
 */
public SpeciesNames obtainSpeciesNames();

/** Obtains local naturalists contact details.
 *
 *  Creates an instance of Naturalist,
 *  if local XML file is available uses method
 *  inside this instance to parse the content.
 *  Else calls setNaturalistDetails() passing the naturalist object.
 *  @return Naturalist naturalist new instance of Naturalist created.
 */
public Naturalist obtainNaturalist();

/** Sets contact details for naturalist.
```


Copyright Aberystwyth University / Computer Science

```
     *
     *  Takes user input to set contact details for the naturalist provided.
     *  @param Naturalist naturalist sets contact details for this object.
     */
    public void setNaturalistDetails();

    /** Checks for updates of the reserves and species data.
     *
     *  Takes local data versions and calls their respective methods
     *  to check if there are updates.
     *  If updates are available then calls their respective methods
     *  to download the new version and parse it to an XML file,
     *  to store it on Android.
     */
    public void checkUpdates();

    /** Creates a visit instance.
     *
     *  Creates a new visit using the naturalists contact details..
     */
    public void recordVisit();

    /** Uploads visit to server.
     *
     *  Prepares visit to be send to the server when data network is available
     *  by calling a method in visit instance to parse the object
* to an XML file.
```


Copyright Aberystwyth University / Computer Science

```
        * The file is then passed to the sendVisit instance created in main,
        * that prepares it to be uploaded.
        */
        public void sendVisit();
}
```

4.2 Send Interface Description

```
public interface SendData{

/*
* This method takes the data from each record in the visit recording and adds it to a
* XML String.
* Calls setPictureContent(int speciesNumber) for each recording if imageTaken field is true.
* @param The visit record.
*/
public void setSendInformation(Data recordData);

/*
* This method takes the processed text data from the full visit record and stores it in a text/xml MIME
* body part object.
*/
public void setTextContent(String recordXML);

/*
* This method should add the image associated with the given record to a image/jpg MIME
* body part object.
*/
```


Copyright Aberystwyth University / Computer Science

```
public void setPictureContent(int speciesNumber);

/*

* This method builds the MIME message in preparation to be sent. It is very important

* that the text portion is added to the message before the image portion due to the

* MIME standard.

*/

public void buildMime();

/*

* This method sets the target recipient of the message and sends it.

*/

public void sendMessage();
```

4.3 Visit Interface Description

```
/** Interface for the Plant Catalog.

+ * Implements parcelable

+ */

+public interface VisitInterface {

+

+ public String searchReserves(String reserves);

+

+ /**

+ * Uses searchReserves() to find reserve names in the database

+ * If reserve does not exist, allows user to use their own reserve

+ * @return string of the reserve name

+ */
```


Copyright Aberystwyth University / Computer Science

```
+ public String obtainReserveName();
+
+ /**
+ * Creates a new plant to be added to the visit
+ * @return plant object
+ */
+ public Plant obtainPlant();
+
+ /**
+ * Automatically obtains the time from the Android clock
+ * @return time string
+ */
+ public String obtainTime();
+}
```

4.4 Plant species Interface Description

```
/** Interface for the Plant Catalog.
 * Implements parcelable
 */
 public interface PlantSpeciesInterface {
 public String searchPlant(String plantName);
 /**Returns XML address of photo
 * If photo is from camera, accessCamera();
 * else accessGallery();
```


Copyright Aberystwyth University / Computer Science

```
 */
 public String obtainPlantPhoto();
 /**
 * Accesses the camera of the mobile phone
 * @return photo form camera
 */
 public String accessCamera();
 /**
 * Accesses the gallery of the mobile phone
 * @return photo form gallery
 */
 public String accessGallery();
 /**
 * Uses searchPlant() to find plant names in the database
 * If plant does not exist, allows user to use their own species name
 * @return string of the plant name
 */
 public String obtainPlantName();
 /**
 * Obtains GPS location from Android
 * @return the GPS loctaion
 */
 public String obtainGpsLocation();
```


Copyright Aberystwyth University / Computer Science

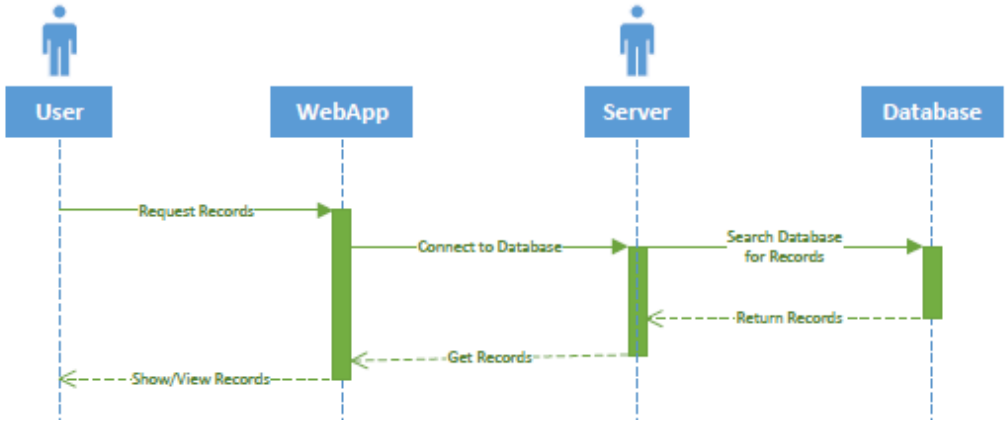
```
/**
 * Obtains the text from the comment text box and adds it to the plant details
 */
public void addComment();
}
```

5 Detailed Design

In the detail design the group should provide information about how the difficult bits of the design and how all the classes work together. Also for every module that is not self-evident details should be provided in order to demonstrate the feasibility and coherence of the overall design. UML diagrams are a good way of documenting how the classes work together.

5.1 Sequence Diagrams

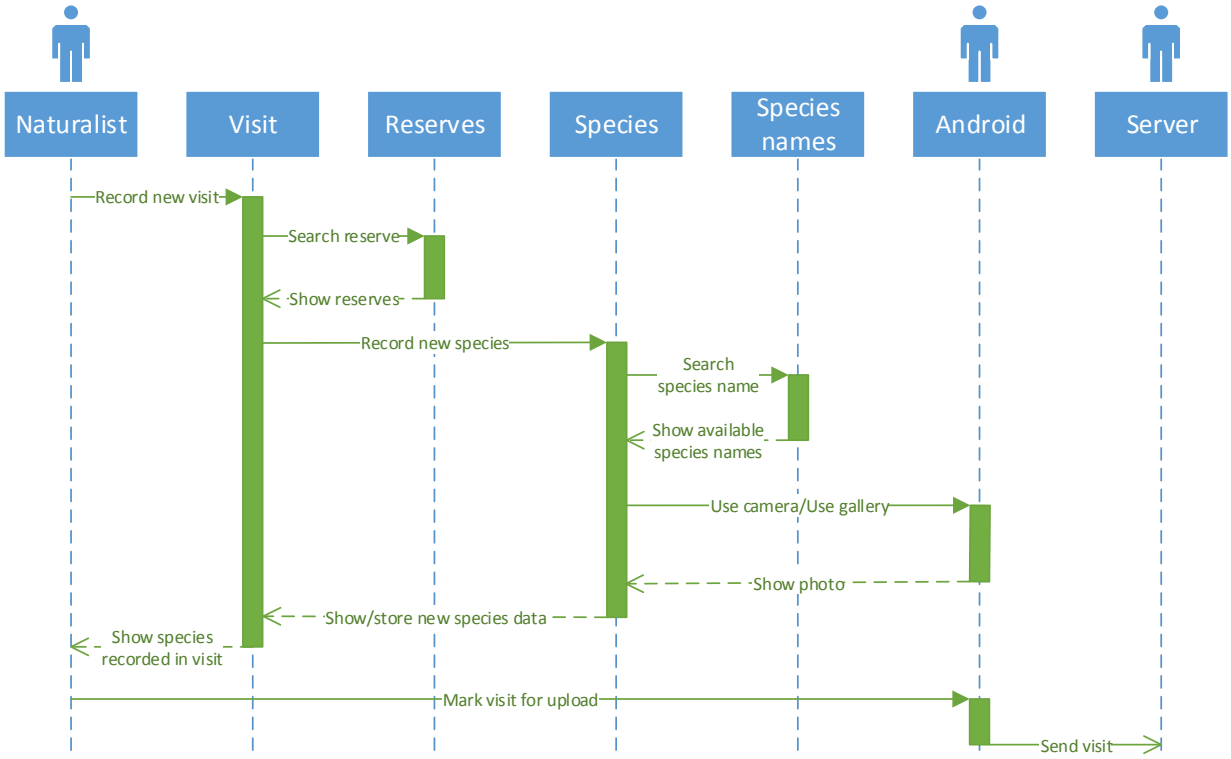
5.1.1 Web Sequence Diagrams



Copyright Aberystwyth University / Computer Science

The user request data from the server through the web app which connects to the server. After getting this command the server will look in the database for the required data. Once this is done the records are being returned from the database in the server. The web app then gets the records from the server and returns them to the user for viewing.

5.1.2 Java Sequence Diagrams



This is a very simple representation of the sequence a user/naturalist undergoes to record a visit and a species. Once the data for the visit has been provided in text form, including the location gotten from the reserves, a species can be recorded. Data for the species can be provided in text form as well as from the species names. Optionally the camera of the phone or the photo gallery can be accessed to include up to two photos for each species.

Once the recording has finished, the naturalist has the option to mark the visit as sent. This parses the visit object with its entire species in an XML file on Android. It then waits for network connection so it can be uploaded to the server. This is done in a separate thread so the naturalist doesn't have to wait.

5.2 Significant Algorithms

These algorithms are shown here are the most important ones that will be used they are also the ones that need the most thought put in to them so we can fully understand what is going to be happing. So should something go wrong or error there will be more than one person that can help to fix the error.

Copyright Aberystwyth University / Computer Science

5.2.1 Significant Web Algorithms

The following algorithm retrieves all of the recordings that have been stored in the database and displays them in a table format with appropriate column headers on the web page, with each row of the table showing 1 recordings data.

connect to database

result = query database -> fetch all recordings

FOREACH(array_keys(fetch assoc(result)) as column_name) {
output column_name as table header
}

free result(result)
result = query database -> fetch all recordings
WHILE(row = fetch assoc(recordings)) {
output recording data as table row
}

Copyright Aberystwyth University / Computer Science

Reserves Page Algorithm

The following algorithm retrieves all of the reserves that have been stored in the database and displays them in a table format with appropriate column headers on the web page, with each row of the table displaying the information about 1 reserve as well as an option to edit or delete the reserve.

```
connect to database

result = query database -> fetch all reserves
FOREACH(array_keys(fetch assoc(result)) as column_name) {
output column_name as table header
output table headers for edit and delete
}

free result(result)
result = query database -> fetch all reserves
WHILE(row = fetch assoc(reserves)) {
output reserve data as table row
output button to edit reserve
output button to delete reserve
}

onClick edit reserve
redirect to edit reserve page
onClick remove reserve
query database -> remove reserve
```

5.1.2 Significant Android Algorithms

Program Launch

User -> Presses on app launcher.
New activity started.

```
checkNewDatabase()
        if (newDatabase  == true){
                getDatabaseXMLFromServer();}
        databaseXML = getDatabaseXMLFromLocal();}
loadGUI();
checkPreviousUser()
        if (!user.exists){
                newUserPrompt();}
loadUser(), loadDatabase(), newVisit();
```

The above method describes how the Android application launches. First it checks if there is a new database version available from the web server using the method described below. Then, once it has

Copyright Aberystwyth University / Computer Science

retrieved the database (assuming there is a new version available) and saved it locally, it loads the GUI and checks if there has been a previous user login.

If there has been no previous user, it prompts for a new user using the method described below. Once done, or if there is a user saved already, it loads the database and prompts the user for a new visit.

Database Handling

```
boolean checkNewDatabase(){
        INT webDatabaseVersion = server.databaseVersion
        INT currentDatabaseVersion = local.getDatabaseVersion
        if (currentDatabaseVersion < webDatabaseVersion){
                return true;}
        else{
                return false;}
```

This method simply compares the current local database version with the one found on the server. if the server's version is newer (that is, larger than the current version), then it returns True that there is a new version of the database. Else it returns false.

```
void getDatabaseXMLFromServer(){
        FileOutputStream.getChannel().transferFrom(WEBSERVER);}
```

After the above, this is called, simply downloading the XML file straight to the android handset in the desired location.

Creating a User

```
PRINT "New User Name";
STRING userName = scanner.nextln();
//Repeat the above for phone and email
FileOutputStream.saveXML(userData);
```

This algorithm saves the user details file to XML for loading later.

Taking Recording From User

For the most part the taking of a recording is handled by the GUI, with strings saved from the text input once the user saves the record. What is automatic is the collection of the GPS location, which can be taken as a snapshot using the Android's location service, and the image file switch, which is a simple boolean indicating that at least one picture has been taken, followed by an int showing the number of total pictures taken.

Saving Input From User

```
STRING userEntry; //The options added to the record for this recording.
if (recording.hasPicture == TRUE){
userEntry = userEntry + pictureCount
FileOutputStream(visitXML) = visitXML + userEntry;
```


Copyright Aberystwyth University / Computer Science

```
RETURN returnPlants;}
```

For the most part, the user will be filling in details based on text boxes. What is automatic is the gathering of the current location. The above algorithm takes everything the user has entered, adds the GPS data, sets whether a photograph has been taken or not, then saves it to a new line in the visit's XML file.

Creating and Sending the MIME Message

```
MIME textPart = new MIMEBodyPart();
textPart.setContent (recordData, XML);
MultiPart.addBodyPart(textPart);

MIME imgPart = new MIMEBodyPart();
if (recordData.imageCount > 0){
        for(recordData.imageCount){
                imgPart.setContent(image);
                MultiPart.addBodyPart(imgPart);}}

MIME.setContent(MultiPart);}
```

The above algorithm shows how the XML text portion of the data will be added to a MIME message, followed by any images recursively. The images must go after the XML due to an operating standard within the MIME protocol.

```
MIME.setRecipient(server);
MIME.setSubject(recordName);
MIME.setSentDate(new Date());
addToSendStack(MIME);

VOID sendStack(){
        while(itemsOnStack){
                if(internetConnection = true){
                        while(internetConnection)
                                for(itemsOnStack){
                                        Transport.send(MIME);}}
                wait(60seconds);
```

The rest of this algorithm adds the destination information to the MIME message, then adds it to a stack. This is a loop on its own thread (current design) that, if there are files to send, checks if there is an internet connection every 60 seconds. Once there is, it will upload any visit files on the stack until either the internet cuts out, or it runs out of files.

Copyright Aberystwyth University / Computer Science

Authors: Christopher Malton (cpm6), Jonathon Shire (jos56), Mohannad Zeido (moz1), Sophie Joseph (soj6), Christi Toba (sit10), Awais Ahmed (awa), Peter Newbold (pjn), Andreas Hernandez (anh45), Rhydian Jenkins (rlj10), Nicholas Gray (nig13)

Version	Date	Changes Made	Changed by
1.0	01.12.2014	Original document.	cpm6
1.1	02.12.2014	Added two more interfaces.	cpm6
1.2	02.12.2014	Made some changes in accordance with the formal review.	cpm6

Copyright Aberystwyth University / Computer Science