# CONTENTS

Purpose of this document 2

Scope 2

Objectives 2

Application Description 2

RPS recorder 2

Program Structure 2

RPSRecord 2

RPS View 3

RPSView/public\_html 3

Algorithms 4

RPSview 4

Main Data Areas 5

File system interaction 5

RPS Recorder 5

RPS View 5

Improvement Suggestions 5

RPS Record 5

RPS View 5

Physical limitations 6

RPS Record 6

Rebuilding and testing 6

Document history 6

## Purpose of this document

## This document serves the purpose of making the maintenance of the RPSRecord and RPS View easier by helping the maintainer easily figure out what the program does , how it does it and how they can carry out quick, efficient maintenance on the program.

## Scope

This document includes an overview of the system, the basic structure and algorithms , limitations and suggestions for when making changes.

## Objectives

The main objectives of this document are to allow the program code we have created to be more easily maintained by people carrying out further maintenance on it , helping them to mitigate undetected bugs, add additional features to improve functionality and/or change the program so that changes in external requirements are met.

# Application Description

## RPS recorder

This application can record a plant , add information of that species ,such as DAFOR scale ,comments m location photots, and upload recording to the server

# Program Structure

## RPSRecord

The program is structured in the following classes:

1.MainView - The layout that contains the NewRecordFragment and

2.MainActivity - The activity that holds the main page.

3.MyTabListener - Handles switching between the tabs in MainView.

4.NewRecordFragment - The fragment tab that lets the user enter the details of a new record.

5.RecordViewFragment - The fragment tab that lists all records recorded on the device.

6.UserDataView - The activity that lets the user input their personal information.

7.GPSToGrid - Converts GPS coordinates to an OS grid reference using an external library.

9.InvalidFieldException - An exception used when a field is entered with invalid information.

11.DatabaseHelper - Manages the local database for records and reserves.

12.Record - An object representing a single plant record.

13.Reserve - An object representing a single reserve.

14.ReserveDataManager - Pulls reserve names from the server and stores them locally.

16.SubmitRecord - Handles submitting records to the server database.

17.UserInfo - Holds the user's information in public static variables so that they can be accessed easily from other parts of the program.

The program also contains classes that no longer are in use.

## RPS View

### RPSView/public\_html

Add\_record.php

Takes JSON data sent from the app and adds the recording data to the database.

Add\_reserve.php

Code to validate the user’s inputted data when adding a new reserve, and if that data is valid adding that data as a new reserve in the database.

Connect.php

Database connection variables used on every page, so extracted to it’s own php file that is included on other pages so it doesn’t have to be re-typed every time.

Edit\_list.php

Displays the Edit Reserve List page, with a table of all the reserves and checkboxes to select the reserves the user wants to delete. Also a small piece of JavaScript to warn the user when they select a reserve that has associated recordings selected, as those recordings will also be deleted.

Header.php

The standard header shown on every page.

Index.php

Displays the main homepage - pulls the list of reserves from the database and displays it in a table. It also has code included from add\_reserve.php to add a new reserve and validate the user’s input.

Json.reserves.php

Gets a reserve data and outs it into a Json

Records.php

Displays the recordings saved at a specific reserve, formatted in a table.

Update\_record.php

Displays a page to let the user update the details of a reserve, with JavaScript validation to check the user has entered valid data.

# 

# Algorithms

## RPS view

User Input Validation

function check\_input(){

var gridRegExp = /([a-zA-Z]{2})+([0-9]{6})/; //Two letters followed by 6 numbers

var textRegExp = /[a-zA-Z 0-9]/; //Letters and numbers

if(textRegExp.test(document.details.reserve\_name.value)

&& gridRegExp.test(document.details.grid\_reference.value)

&& textRegExp.test(document.details.description.value))

{

document.details.setAttribute("method", "post");

document.details.setAttribute("action", "add\_reserve.php");

}

else if(!textRegExp.test(document.details.reserve\_name.value)){

alert("Invalid Reserve Name details entered");

}

else if(!gridRegExp.test(document.details.grid\_reference.value)){

alert("Invalid grid reference entered, use a 6 figure OS grid reference" +

" (ie, two letters then 6 numbers");

}

else if(!textRegExp.test(document.details.description.value)){

alert("Invalid description entered");

}

}

This code is from add\_reserve.php and is JavaScript to validate the user’s input, and if the user has entered valid data to set that input to be posted so it can be added to the database.

It first sets up two regular expressions for comparison - one for standard text (ie the reserve name and the reserve description) and one for the grid reference.

The main code is a series of if statements the first comparing the user’s input with the regular expression. If all three comparisons return true values, the subsequent code to alter the form “details”’s attributes so the data is posted back to add\_reserve.php so other code in the file can add the data to the database.

If the user’s data doesn’t pass the first check, it subsequently checks each individual comparison, and if the comparison returns a false value the user is alerted to their error. Other codes are mentioned in detail in the design documentation.

# Main Data Areas

## Data structures are mentioned in detail in the design documentation.

# File system interaction

## RPS recorder

The program requires access to the file-system. Photos taken are store into phones gallery.

# Improvement suggestions

## RPS Record

We tested when in the add new record page , screen rotating will cause the current record lost, potential place for the bug is in the fragment page. One of the big issues is the images uploading may cause error , the record will be reaching the server but unknown problem cause the database to reject the upcoming record, high chances this will occur when images are included in the recording. Due to the development time is shorter than we expect, we did not have a function to edit records in the Record application . Our software also lack of warning/popup message when recordings been uploaded. There are no button to cancel recording, in the current version of the software the only way to do it is do press return buttons of the phone devices.

## RPSView

Since everyone can edit/delete records in the website we suggest adding a login system , only researchers had the right to access the database. There should be a regular database backup to prevent data lost.To have a better organise structure, create an extra page within each reserve for each species in the reserve, then within each species you have the page for the recordings of that species.

# Physical limitations

## RPS Record

The build target of the Android project is version 4.2.2, meaning other API versions may encounter problems running the app.

# Rebuilding and testing

To rebuild the Android app, import the entire project in Eclipse. To test it, install the generated .apk file on an Android device or use an emulator.

# Document history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Version CCF No.* | | *Date* | *Changes made to document* | *Changed by* |
| **1.0** | N/A | 9/02/15 | First version of the document | Yta |