

Web-Based Music School Manager

Design Manual

Author: Aaron Redmond

Student ID: C00002396

Supervisor: Paul Barry

# Contents

[1 Introduction 8](#_Toc284168163)

[2 Phase 1 – Set up 9](#_Toc284168164)

[2.1 Homepage 10](#_Toc284168165)

[2.1.1 Links 10](#_Toc284168166)

[2.1.2 Data 10](#_Toc284168167)

[2.1.3 Processes 11](#_Toc284168168)

[2.1.4 Pseudo code 11](#_Toc284168169)

[2.2 Register New Account 12](#_Toc284168170)

[2.2.1 Data 12](#_Toc284168171)

[2.2.2 Processes 13](#_Toc284168172)

[2.2.3 Pseudo code 13](#_Toc284168173)

[2.2.4 Users 14](#_Toc284168174)

[2.3 Request User Name/Password 15](#_Toc284168175)

[2.3.1 Data 15](#_Toc284168176)

[2.3.2 Processes 15](#_Toc284168177)

[2.3.3 Pseudo Code 15](#_Toc284168178)

[3 Phase 2 – Admin 16](#_Toc284168179)

[3.1 Admin Home 17](#_Toc284168180)

[3.1.1 Links 17](#_Toc284168181)

[3.2 Manage 18](#_Toc284168182)

[3.2.1 Links 18](#_Toc284168183)

[3.3 Manage School 19](#_Toc284168184)

[3.3.1 Links 19](#_Toc284168185)

[3.3.2 Processes 19](#_Toc284168186)

[3.3.3 Pseudo Code 19](#_Toc284168187)

[3.4 Edit School Information 20](#_Toc284168188)

[3.4.1 Links 20](#_Toc284168189)

[3.4.2 Data 20](#_Toc284168190)

[3.4.3 Processes 20](#_Toc284168191)

[3.4.4 Pseudo Code 21](#_Toc284168192)

[3.5 Manage Rooms 22](#_Toc284168193)

[3.5.1 Links 22](#_Toc284168194)

[3.5.2 Data 22](#_Toc284168195)

[3.5.3 Processes 22](#_Toc284168196)

[3.5.4 Pseudo Code 23](#_Toc284168197)

[3.6 Add Room 24](#_Toc284168198)

[3.6.1 Links 24](#_Toc284168199)

[3.6.2 Data 24](#_Toc284168200)

[3.6.3 Processes 24](#_Toc284168201)

[3.6.4 Pseudo Code 24](#_Toc284168202)

[3.7 Edit Room 26](#_Toc284168203)

[3.7.1 Links 26](#_Toc284168204)

[3.7.2 Data 26](#_Toc284168205)

[3.7.3 Processes 26](#_Toc284168206)

[3.7.4 Pseudo Code 26](#_Toc284168207)

[3.8 Manage Subjects 28](#_Toc284168208)

[3.8.1 Links 28](#_Toc284168209)

[3.8.2 Data 28](#_Toc284168210)

[3.8.3 Processes 28](#_Toc284168211)

[3.8.4 Pseudo Code 29](#_Toc284168212)

[3.9 Edit Subject 30](#_Toc284168213)

[3.9.1 Links 30](#_Toc284168214)

[3.9.2 Data 30](#_Toc284168215)

[3.9.3 Processes 30](#_Toc284168216)

[3.9.4 Pseudo Code 30](#_Toc284168217)

[3.10 Manage Teachers 32](#_Toc284168218)

[3.10.1 Links 32](#_Toc284168219)

[3.10.2 Data 32](#_Toc284168220)

[3.10.3 Processes 33](#_Toc284168221)

[3.10.4 Pseudo Code 33](#_Toc284168222)

[3.11 Add Teacher 34](#_Toc284168223)

[3.11.1 Links 34](#_Toc284168224)

[3.11.2 Data 34](#_Toc284168225)

[3.11.3 Processes 35](#_Toc284168226)

[3.11.4 Pseudo Code 35](#_Toc284168227)

[3.12 Edit Teacher 37](#_Toc284168228)

[3.12.1 Links 37](#_Toc284168229)

[3.12.2 Data 37](#_Toc284168230)

[3.12.3 Processes 38](#_Toc284168231)

[3.12.4 Pseudo Code 38](#_Toc284168232)

[3.13 Manage Students 40](#_Toc284168233)

[3.13.1 Links 40](#_Toc284168234)

[3.13.2 Data 40](#_Toc284168235)

[3.13.3 Processes 41](#_Toc284168236)

[3.13.4 Pseudo Code 41](#_Toc284168237)

[3.14 Add Student 42](#_Toc284168238)

[3.14.1 Links 42](#_Toc284168239)

[3.14.2 Data 42](#_Toc284168240)

[3.14.3 Processes 43](#_Toc284168241)

[3.14.4 Pseudo Code 43](#_Toc284168242)

[3.15 Edit Student 44](#_Toc284168243)

[3.15.1 Links 44](#_Toc284168244)

[3.15.2 Data 44](#_Toc284168245)

[3.15.3 Processes 45](#_Toc284168246)

[3.15.4 Pseudo Code 45](#_Toc284168247)

[3.16 Manage Groups 46](#_Toc284168248)

[3.16.1 Links 46](#_Toc284168249)

[3.16.2 Data 46](#_Toc284168250)

[3.16.3 Processes 47](#_Toc284168251)

[3.16.4 Pseudo Code 47](#_Toc284168252)

[3.17 Add Group 49](#_Toc284168253)

[3.17.1 Links 49](#_Toc284168254)

[3.17.2 Data 49](#_Toc284168255)

[3.17.3 Processes 50](#_Toc284168256)

[3.17.4 Pseudo Code 50](#_Toc284168257)

[3.18 Edit Group 52](#_Toc284168258)

[3.18.1 Links 52](#_Toc284168259)

[3.18.2 Data 52](#_Toc284168260)

[3.18.3 Processes 53](#_Toc284168261)

[3.18.4 Pseudo Code 53](#_Toc284168262)

[3.19 Add Group Type 55](#_Toc284168263)

[3.19.1 Links 55](#_Toc284168264)

[3.19.2 Data 55](#_Toc284168265)

[3.19.3 Processes 55](#_Toc284168266)

[3.19.4 Pseudo Code 55](#_Toc284168267)

[3.20 Manage Lessons 56](#_Toc284168268)

[3.20.1 Links 56](#_Toc284168269)

[3.20.2 Processes 56](#_Toc284168270)

[3.20.3 Pseudo Code 57](#_Toc284168271)

[3.21 Add Lesson 58](#_Toc284168272)

[3.21.1 Links 58](#_Toc284168273)

[3.21.2 Data 58](#_Toc284168274)

[3.21.3 Processes 59](#_Toc284168275)

[3.21.4 Pseudo Code 59](#_Toc284168276)

[3.22 Edit Lesson 60](#_Toc284168277)

[3.22.1 Links 60](#_Toc284168278)

[3.22.2 Data 60](#_Toc284168279)

[3.22.3 Processes 61](#_Toc284168280)

[3.22.4 Pseudo Code 61](#_Toc284168281)

[3.23 Timetabling 62](#_Toc284168282)

[3.23.1 Links 62](#_Toc284168283)

[3.23.2 Data 62](#_Toc284168284)

[3.23.3 Processes 63](#_Toc284168285)

[3.23.4 Pseudo Code 63](#_Toc284168286)

[3.24 Timetabling Information 64](#_Toc284168287)

[3.24.1 Data 64](#_Toc284168288)

[3.24.2 Processes 64](#_Toc284168289)

[3.24.3 Pseudo Code 64](#_Toc284168290)

[3.25 Instruments 66](#_Toc284168291)

[3.25.1 Links 66](#_Toc284168292)

[3.25.2 Data 66](#_Toc284168293)

[3.25.3 Processes 67](#_Toc284168294)

[3.25.4 Pseudo Code 67](#_Toc284168295)

[3.26 Personalisation 69](#_Toc284168296)

[3.26.1 Links 69](#_Toc284168297)

[3.26.2 Data 69](#_Toc284168298)

[3.26.3 Processes 69](#_Toc284168299)

[3.26.4 Pseudo Code 70](#_Toc284168300)

[3.27 News 71](#_Toc284168301)

[3.27.1 Data 71](#_Toc284168302)

[3.27.2 Processes 71](#_Toc284168303)

[3.27.3 Pseudo Code 72](#_Toc284168304)

[3.28 My Account 73](#_Toc284168305)

[3.28.1 Data 73](#_Toc284168306)

[3.28.2 Processes 73](#_Toc284168307)

[3.28.3 Pseudo Code 73](#_Toc284168308)

[4 Phase 3 – Teacher 75](#_Toc284168309)

[4.1 Home 76](#_Toc284168310)

[4.1.1 Links 76](#_Toc284168311)

[4.1.2 Pseudo Code 76](#_Toc284168312)

[4.2 Timetable 77](#_Toc284168313)

[4.2.1 Processes 77](#_Toc284168314)

[4.2.2 Pseudo Code 78](#_Toc284168315)

[4.3 My Account 79](#_Toc284168316)

[5 Phase 4 – Student 80](#_Toc284168317)

[5.1 Student Home 81](#_Toc284168318)

[5.1.1 Links 81](#_Toc284168319)

[5.1.2 Pseudo Code 81](#_Toc284168320)

[5.2 Timetable 82](#_Toc284168321)

[5.2.1 Processes 82](#_Toc284168322)

[5.2.2 Pseudo Code 82](#_Toc284168323)

[5.3 My Account 83](#_Toc284168324)

[6 Appendix A – Timetable Generation 84](#_Toc284168325)

[6.1 A little bit on genetic algorithms 84](#_Toc284168326)

[6.2 Implementation 85](#_Toc284168327)

[7 References 87](#_Toc284168328)

# Introduction

This paper is the design manual for the web based music school manager. The main goal for this project is to build an easy to use piece of software, which will help reduce the time spent on the administration and timetabling tasks within a small music school.

The design of this project can be broken down into four separate phases, as described in the project plan. These phases are:

1. Phase 1 – Set Up
2. Phase 2 – Admin
3. Phase 3 – Teacher
4. Phase 4 – Student

At the end of each phase, we should have a working version of the system, with each version building upon the last.

# Phase 1 – Set up

This phase involves all the details and functionality around user registration and log in. The user logs in on the homepage, and can access both the register new account and request username\password pages from here.

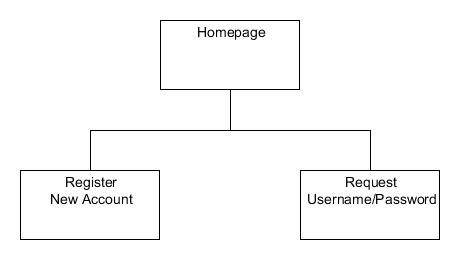


Figure 1: Phase 1

## Homepage



Figure 2: Music School Manager Homepage

The homepage is the first page a user will view when they navigate their browser to the Music School Manager website. The design should be clean and all options should be easy to access for the users.

An existing user may login with their username and password. A new user must first register a new account before they can log into the system. If for whatever reason a user forgets their user name or password they can request their login details to be sent to them via e-mail.

If an incorrect user name or password is supplied then the system should notify the user that they have entered an incorrect value and to please try again.

### Links

This page directly links to the following pages:

* Register New Account
* Request User Name/Password

### Data

User Name

* A user may enter their unique user name into this field for validation. The username can be between 6 and 20 characters and may contain the follow letters, number, and symbols:
* Uppercase letters (A-Z)
* Lowercase letters (a-z)
* Digits (0-9)
* Symbols (“@.\_”)

Password

* A user may enter the password associated with their user name into this field for validation. The password can be between 6 and 20 characters and may contain the following letters, numbers and symbols:
* Uppercase letters (A-Z)
* Lowercase letters (a-z)
* Digits (0-9)
* Symbols (~#@;:[]{}()\*&^%$£”!|\<>,.?/)

### Processes

Login

* Once the user has entered their user name and password, they can click this button to log into the system. If the user has not entered a user name, a password, or both, the system will notify them and highlight the blank field with an asterisk.

Register New Account

* A new user can click this link, which will bring them to the Register New Account page.

Forgot Username/Password

* Any registered user who has forgotten either their user name or password can click this link which will bring them to the request user name/password page.

### Pseudo code

When the user clicks the log in button the following processes should occur:

* Validate username and password.
  + The system checks the username against users table in the database. If the name is valid then check the password entered against the stored password in the users table.
  + If either check fails then notify the user that they have entered an incorrect username/password.
* If both checks return true then set the log in state to true and navigate to the correct logged in state for the user type.

## Register New Account

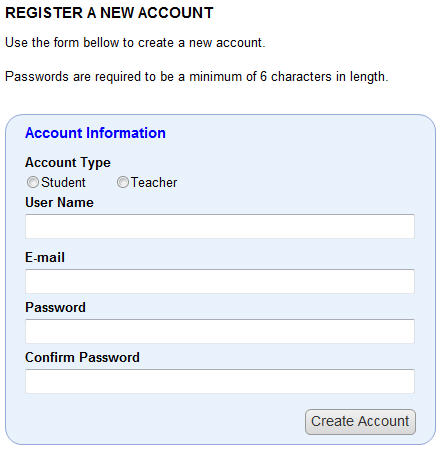


Figure 3: Register New Account

The register new account page, which is accessed via the homepage, is where a new user can create an account. The new user simply selects the account type then fills in their desired user name, e-mail address, password and then confirms their desired password. Once all the information has been entered they may click the Create Account button.

### Data

Account Type

* The user must state their intended account type, which can be either student or teacher. The default setting is for a student account.

User Name

* The same user name rules apply as in the above section 2.1.2.

Password

* The same password rules apply as in the above section 2.1.2.

E-mail

* An e-mail address such as [c00002386@itcarlow.ie](mailto:c00002386@itcarlow.ie) is comprised of two parts. The part before the @ is the local part of the address, the recipient (c00002396), and the part after the @ is a domain name to which the e-mail will be sent (itcarlow.ie).
* The local part of the e-mail address may use any of these characters:
* Uppercase and lowercase letters (a–z, A–Z)
* Digits (0 to 9)
* Characters (! # $ % & ' \* + - / = ? ^ \_ ` { | } ~)
* Character (. period) provided that it is not the first or last character, and provided also that it does not appear two or more times consecutively.

Confirm Password

* This field is identical in description to the password field, however the password entered here should match the previously entered password.

### Processes

Create Account

* Once the user has filled in the form, they can click this button to submit their detail. If any of the data fields has been left blank then the system should notify the user and highlight the blank fields with an asterisk.

### Pseudo code

As the page loads the following process should occur:

* The system automatically selects the student account type; this is the default account type.

When a user clicks on the create account button, the following validation processes are carried out:

* Ensure all fields are filled in. If any field has been left blank then notify the user that all fields must be filled in and highlight any blank fields with an asterisk.
* Ensure that their username is unique, by checking it against the existing usernames in the users table of the database.
* Ensure that the e-mail address is valid. This can be done by checking the e-mail address entered against the following regular expression or something similar:
  + '/^\S+@[\w\d.-]{2,}\.[\w]{2,6}$/iU'
* Ensure that the password entered in the password field matches the password entered in the verify password field.

If the validation process returns true then the following create account processes occur:

* The system creates the new user record in the users table and sets the user\_verified flag to false.
* The system then alerts the administrator, who then either accepts or rejects the new account.
* If the administrator accepts the new account, the system sets the user\_verified flag to true and notifies the user via e-mail.
* If the administrator rejects the account, the system deletes the new account from the users table and notifies the user via e-mail.

### Users

There will be three different user groups:

1. Admin
   * There will be only one administrator user.
2. Teacher
   * A user within this group will have a teacher account.
3. Student
   * A user within the group will been given a student account. This group will be the largest user group.

## Request User Name/Password

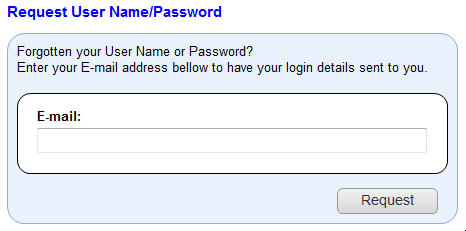


Figure 4: Request Username/Password

The request user name/password page is accessible via the homepage. It provides users who have forgotten their user name or password a means by which they can enter the e-mail address they registered with and get their login information sent to them.

### Data

E-mail

* The e-mail address entered here should conform to the definition specified in the previous section 2.2.1.

### Processes

Request

* This process sends the requested user name and password to the e-mail address specified in the form.

### Pseudo Code

The following is some pseudo code to represent what happens when the user clicks the request button.

* Check to ensure the e-mail field is not blank.
* Check to ensure the e-mail address is a valid e-mail address. See section 2.2.3.
  + Check the users table in the database to see if there is a user with the specified e-mail address. If yes then retrieve the associated username and password and e-mail retrieved username and password to supplied e-mail address.
  + If no then alert user than the e-mail address does not exist within the system.

# Phase 2 – Admin

Phase 2 is involved with implementing the admin side of the application. The admin section is by far the largest section of the project containing the most functionality.

## Admin Home



Figure 5: Admin Home

This is the first page an admin will see when they successfully log in. From here they can access all of the functionality available to an administrator.

### Links

This page directly links to the following pages:

* Manage
* Timetabling
* System Setup
* My Account

## Manage



Figure 6: Manage page

This is the manage main page, from here the user can access all of the manage pages.

### Links

This page directly links to the following pages:

* Manage School
* Manage Rooms
* Manage Subjects
* Manage Teachers
* Manage Students
* Manage Groups
* Manage Lessons

## Manage School



Figure 7: Manage School

This page allows the user to view the schools information. The school information consists of:

* Name
* Address
* Telephone
* E-mail

### Links

This page directly links to the following pages:

* Edit School Information

### Processes

Edit

* Clicking on the edit button opens the edit school information page.

### Pseudo Code

As the page loads the following process should occur:

* The system retrieves the school information from the schoolinfo table in the database, and populates the forms data fields.

## Edit School Information

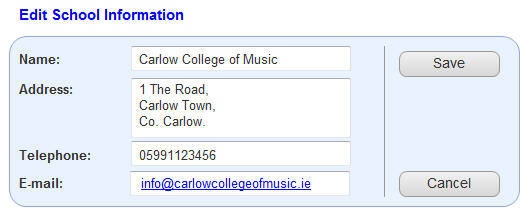


Figure 8: Edit School Information

This page allows the user to edit the schools information.

### Links

This page directly links to the following pages:

* Manage School

### Data

Name

* The name of the school as entered by the user.

Address

* The address of the school as entered by the user.

Telephone

* The telephone number for the school as entered by the user.

E-mail

* The e-mail address entered here should conform to the definition specified in the previous section 2.2.1.

### Processes

Save

* The system saves the edited information and returns the user to the manage school page.

Cancel

* The edited school information is discarded and the user is returned to the manage school page.

### Pseudo Code

When the save button is clicked, the following processes should occur:

* Ensure the following fields are not blank:
  + Name
  + Address
  + E-mail
* If the telephone field is filled in ensure it contains a valid phone number.
* Validate the e-mail address. See section 2.2.3.
* Update the schoolinfo table in the database with edited information.

When the cancel button is clicked, the following process should occur:

* The system returns the user to the manage school page.

## Manage Rooms

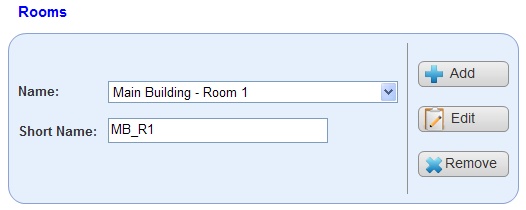


Figure 9: Manage Rooms

This page allows the user to view the rooms available to the school.

### Links

This page directly links to the following pages:

* Add Room
* Edit Room

### Data

Name

* This is a drop down list from which the user can choose a room.

Short Name

* This is a short name for the room; ideally it should be a shortened version of the room name.

### Processes

Add

* Clicking on the add button opens the add room page, when the user can add additional rooms.

Edit

* Clicking on the edit button opens the edit room page, where the user can edit the currently selected school.

Remove

* Clicking on the remove button deletes the currently selected room.

### Pseudo Code

As the page loads the following process should occur:

* The system retrieves the rooms from the rooms table in the database, and populates the name list.
* The first name in the list is selected.
* The system retrieves the correct short name for the selected room.

When the user clicks the add button, the following processes should occur:

* The system navigates to the add room page.

When the user clicks the edit button, the following processes should occur:

* The system remembers the room id.
* The system navigates to the edit room page, and populates its data fields with the values associated with room id.

When the user clicks the remove button, the following processes should occur:

* The system prompts the user “Are you sure you wish to delete this room? Yes/No”.
* If the user responds yes then delete the room from the rooms table in the database and repopulate the name list.
* If the user responds no then no action is taken.

## Add Room



Figure 10: Add Room

If the user clicks the add room button on the manage rooms page it will bring them to this page, the add room page.

### Links

This page links directly back to the following page:

* Manage Rooms

### Data

Name

* The new room name.

Short Name

* The shortened version of the new room name.

### Processes

Save

* Clicking the save button saves the new room information.

Cancel

* Clicking the cancel button, discards any information entered since the last save and returns the user to the manage rooms page.

### Pseudo Code

When the user clicks the save button, the following processes should occur:

* The system ensures that none of the fields are blank. If they are then alert the user.
* The system ensures that both the name and short name are unique, if not then alert the user.
* The system adds the new room information to the rooms table in the database.

When the user clicks the cancel button, the following processes should occur:

* If both fields are blank then return the user to the manage rooms page.
* If either field is non-blank the alert user “Are you sure you wish to navigate away from this page? Yes/No”. If yes then return the user to the manage rooms page, if no then no action is taken.

## Edit Room

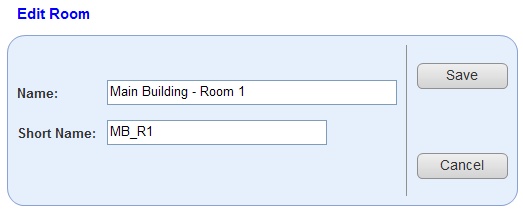


Figure 11: Edit Room

If the user clicks the edit room button on the manage rooms page it will bring them to this page, the edit room page.

### Links

This page links directly back to the following page:

* Manage Rooms

### Data

Name

* The room name to edit.

Short Name

* The shortened version of the room name to edit.

### Processes

Save

* Clicking the save button saves the edited room information.

Cancel

* Clicking the cancel button, cancels the edit and returns the user to the manage rooms page.

### Pseudo Code

When the user clicks the save button, the following processes should occur:

* The system ensures that none of the fields are blank. If they are then alert the user.
* The system ensures that both the name and short name are unique, if not then alert the user.
* The system updates the edited room information in the rooms table, and returns the user to the manage rooms page.

When the user clicks the cancel button, the following processes should occur:

* Alert user “Are you sure you wish to cancel the edit? Yes/No”. If yes then return the user to the manage rooms page, if no then no action is taken.

## Manage Subjects

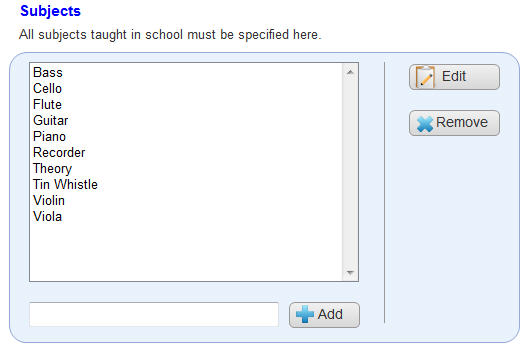


Figure 12: Manage Subjects

From this page the user can view, edit or remove existing subjects. They may also add additional subjects. Initially the first item in the subjects list is highlighted.

### Links

This page directly links to the following pages:

* Edit Subject

### Data

Subjects

* This is the list of subjects taught at the school.

New Subject

* This is new subject the user wishes to add to the subjects list.

### Processes

Add

* The user enters a new subject and clicks the add button. The system adds the new subject to the subjects list.

Edit

* The system navigates to the edit subject page, where the selected subject may be edited.

Remove

* Removes the selected subject.

### Pseudo Code

When the page loads, the following process should take place:

* The system retrieves the lists of subjects from the subjects table in the database and populates the subject list.

When the user clicks the add button the following processes should take place:

* Ensure the new subject field is not bank, if it is blank alert the user.
* Check the subjects table in the database to see if the new subject already exists.
* If the subject to be added does not exist then add it to the subjects table in the database and update the subject list.
* If the subject to be added exists, notify the user that the subject already exists.

When the user clicks the remove button, the following processes should take place:

* The system prompts the user “Do you really wish to delete this subject? Yes/No”.
* IF the user clicks yes then the system removes the selected subject from the subjects table in the database and updates the subject list.
* If the user clicks no then take no action.

## Edit Subject



Figure 13: Edit Subject

On this page the user can edit the subject which they selected from the manage subjects page.

### Links

This page links back to the following page:

* Manage Subjects

### Data

Subject

* This is the subject to be edited.

### Processes

OK

* The user clicks this button to save the edited subject, and returns to the manage subjects page.

Cancel

* The user clicks this button to discard any changes to the subject, and returns to the manage subjects page.

### Pseudo Code

As this page loads the following process occurs:

* The system fills the subject data field with the subject to be edited.

When the OK button is clicked the following processes occur:

* The subject field is checked to ensure it is not blank, if it is alert the user.
* Check the subjects table in the database to see if the edited subject already exists, if it does alert the user.
* If the edited subject does not exist then update the subjects table in the database and return the user to the manage subjects page.

## Manage Teachers



Figure 14: Manage Teachers

From this page the user can view the currently registered teachers. They may also add a new teacher or edit an existing one.

### Links

This page directly links to the following pages:

* Add Teacher.
* Edit Teacher.

### Data

Name

* This is the list of current teachers. The user may select a teacher from the list box and the system displays the relevant teacher information. The following is the list of information displayed:
  + Address
  + Telephone
  + Mobile
  + Email
  + Subjects taught

### Processes

Add

* The system navigates to the add teacher page, where the user can add one or more new teachers.

Edit

* The system navigates to the edit teacher page, where the selected teacher’s details are available for editing.

Remove

* Remove the selected teacher.

### Pseudo Code

When the page loads, the following process should take place:

* The system retrieves all users from the users table in the database which have the user\_type teacher.
* The first teacher is automatically selected.
* The system retrieves all relevant information from the users table in the database for the selected teacher.

When a different teacher is selected, the following process should take place:

* The system retrieves all relevant information from the users table in the database for the selected teacher.

When the user clicks the remove button, the following processes should take place:

* The system prompts the user “Do you really wish to delete this teacher? Yes/No”.
* If the user clicks yes then the system removes the selected teacher’s record from the users table in the database and updates the teacher list.
* If the user clicks no then take no action.

## Add Teacher

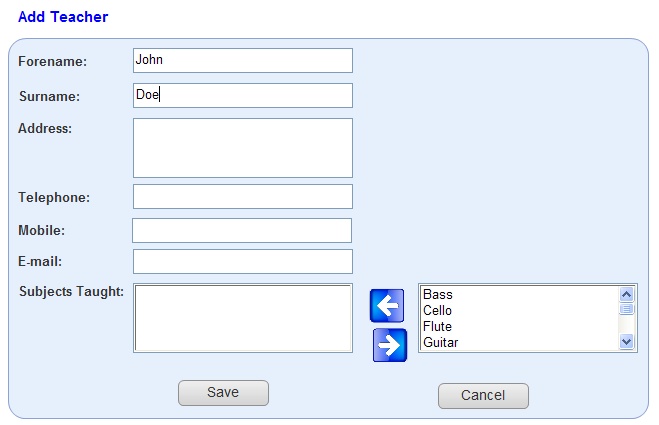


Figure 15: Add Teacher

If the user clicks the add teacher button on the manage teachers page it will bring them to this page, the add teacher page.

### Links

This page links directly back to the following page:

* Manage Teachers

### Data

Forename

* The new teachers forename.

Surname

* The new teachers surname.

Address

* The new teachers address.

Telephone

* The new teachers telephone number.

Mobile

* The new teachers mobile phone number.

E-mail

* The e-mail address entered here should conform to the definition specified in the previous section 2.2.1.

Subject taught

* The subjects the new teacher will be teaching.

Subjects

* The master list of subjects.

### Processes

Save

* The system saves the new teacher.

Cancel

* The system cancels the addition of the new teacher.

Add Subject

* Adds the selected subject to subjects taught.

Remove Subject

* Remove the selected subject from subjects taught.

### Pseudo Code

As this page loads, the following processes should occur:

* The system retrieves the list of subjects from the subjects table in the database and populates the subjects list.
* The first subject in the list is automatically selected.

When the user clicks the add subject button, the following process should occur:

* The selected subject is added to subjects taught.
* The selected subject is removed from the subject master list.

When the user clicks the remove subject button the following process should occur:

* The selected subject is removed from taught subjects.
* If the taught subject data field is empty then no action is taken.
* The selected subject is added to the subject master list.

When the user clicks the save button the following processes should occur:

* Ensure no field are left blank. If any fields have been left blank the alert the user and highlight the blank fields with an asterisk.
* Validate e-mail address. See section 2.2.3.
* Once all fields are validated :
  + Add the new teacher to the users table in the database.
  + Add the subjects taught to the subjectstaught table in the database.
  + Return the user to the manage teachers page.

## Edit Teacher

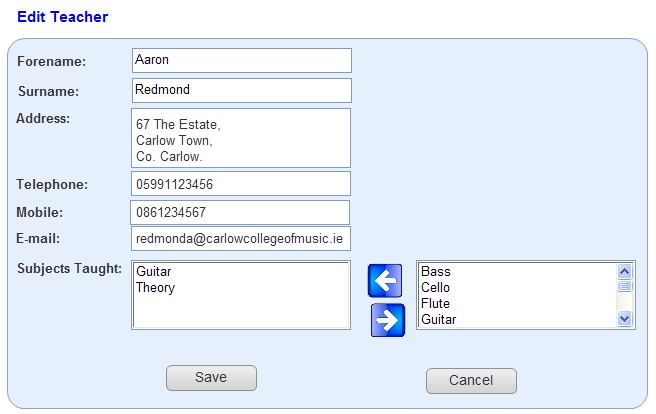


Figure 16: Edit Teacher

If the user clicks the edit teacher button on the manage teachers page it will bring them to this page, the edit teacher page. On this page they can edit any of the details belonging to the selected teacher.

### Links

This page links directly back to the following page:

* Manage Teachers

### Data

Forename

* The forename of the teacher to edit.

Surname

* The surname of the teacher to edit.

Address

* The address of the teacher to edit.

Telephone

* The telephone number of the teacher to edit.

Mobile

* The mobile phone number of the teacher to edit.

E-mail

* The e-mail address entered here should conform to the definition specified in the previous section 2.2.1.

Subject taught

* The subjects the edited teacher will be teaching.

Subjects

* The master list of subjects.

### Processes

Save

* The system saves the edited teacher information.

Cancel

* The system cancels the edit.

Add Subject

* Adds the selected subject to subjects taught.

Remove Subject

* Remove the selected subject from subjects taught.

### Pseudo Code

As this page loads, the following processes should occur:

* The system populates the fields with the data to be edited based on the teacher id passed in.
* The system retrieves the list of subjects from the subjects table in the database and populates the master subject list.
* The system removes any subject in subjects taught from the master subject list.
* The first subject in the master list is automatically selected.

When the user clicks the add subject button, the following process should occur:

* The selected subject is added to subjects taught.
* The selected subject is removed from the subject master list.

When the user clicks the remove subject button the following process should occur:

* The selected subject is removed from taught subjects.
* If the taught subject data field is empty then no action is taken.
* The selected subject is added to the subject master list.

When the user clicks the save button the following processes should occur:

* Ensure no field are left blank. If any fields have been left blank the alert the user and highlight the blank fields with an asterisk.
* Validate e-mail. See section 2.2.3.
* Once all fields are validated update the users table in the database with the selected teachers information and return the user to the manage teachers page.

## Manage Students

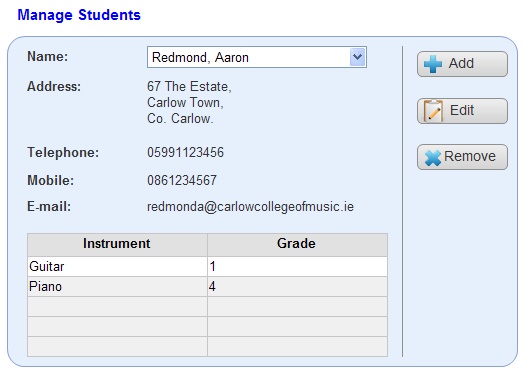


Figure 17: Manage Students

From this page a user can view a student’s details. They may also add a new student or edit an existing one.

### Links

This page directly links to the following pages:

* Add Student.
* Edit Student.

### Data

Name

* This is the list of current students. The user may select a student from the list box and the system displays the relevant student information. The following is the list of information displayed:
  + Address
  + Telephone
  + Mobile
  + Email
  + Instrument/Grade

### Processes

Add

* The system navigates to the add student page, where the user can add one or more new students.

Edit

* The system navigates to the edit student page, where the selected student’s details are available for editing.

Remove

* Remove an existing student.

### Pseudo Code

When the page loads, the following process should take place:

* The system retrieves all users from the users table in the database which have the user\_type student.
* The first student is automatically selected.
* The system retrieves all relevant information from the users table in the database for the selected student.
* The system retrieves the correct instruments and grades via the studentgrades table in the database.

When a different student is selected, the following process should take place:

* The system retrieves all relevant information from the users table in the database for the selected student.
* The system retrieves the correct instruments and grades via the studentgrades table in the database.

When the user clicks the remove button, the following processes should take place:

* The system prompts the user “Do you really wish to delete this student? Yes/No”.
* If the user clicks yes then:
  + The system removes the selected student from the users table in the database and updates the student list.
  + The system removes the instrument and grade records from the studentgrades table in the database.
* If the user clicks no then take no action.

## Add Student

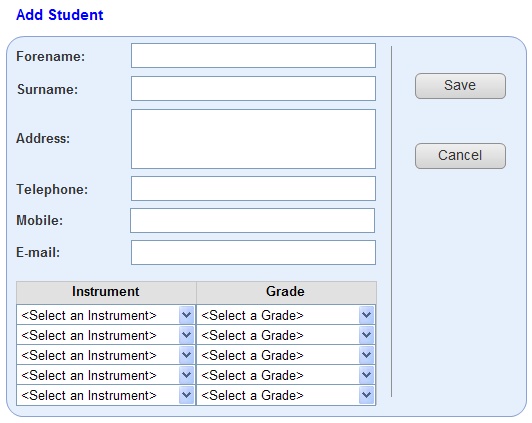


Figure 18: Add Student

If the user clicks the add student button on the manage student page it will bring them to this page, the add student page.

### Links

This page links directly back to the following page:

* Manage Students

### Data

Forename

* The new student forename.

Surname

* The new student surname.

Address

* The new students address.

Telephone

* The new students home telephone number.

Mobile

* The new students mobile number.

E-mail

* The e-mail address entered here should conform to the definition specified in the previous section 2.2.1.

Instrument/Grade

* The instrument and its corresponding grade. A student may have up to 5 instruments at any one time.

### Processes

Save

* The system saves the edited student information.

Cancel

* The system cancels the edit.

### Pseudo Code

As this page loads, the following processes should occur:

* The system retrieves the list of instruments from the instruments table in the database and populates the Instrument drop down lists.
* The system retrieves the list of grades from the grades table in the database and populates the grade drop down lists.

When the user clicks the save button the following processes should occur:

* Ensure no field are left blank. If any fields have been left blank the alert the user and highlight the blank fields with an asterisk.
* Validate the e-mail address. See section 2.2.3.
* Once all fields are validated
  + Add the new student to the users table in the database.
  + Add the new instruments and grades to the studentgrades table in the database.
  + Return the user to the manage students page.

## Edit Student

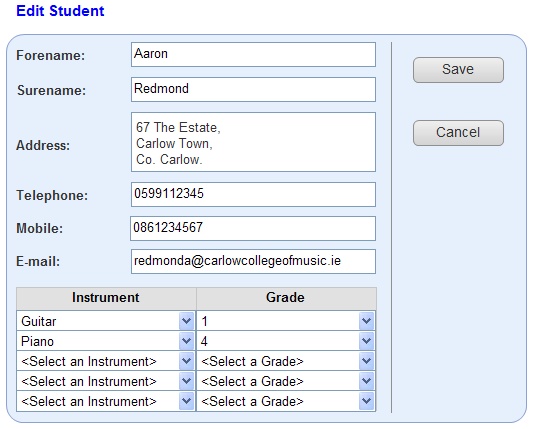


Figure 19: Edit Student

If the user clicks the edit student button on the manage students page it will bring them to this page, the edit student page.

### Links

This page links directly back to the following page:

* Manage Students

### Data

Forename

* The edited students forename.

Surname

* The edited students surname.

Address

* The edited students address.

Telephone

* The edited students home telephone number.

Mobile

* The edited students mobile number.

E-mail

* The e-mail address entered here should conform to the definition specified in the previous section 2.2.1.

Instrument/Grade

* The instrument and its corresponding grade. A student may have up to 5 instruments at any one time.

### Processes

Save

* The system saves the edited student information.

Cancel

* The system cancels the edit.

### Pseudo Code

As this page loads, the following processes should occur:

* The system retrieves the list of instruments from the instruments table in the database and populates the Instrument drop down lists.
* The system retrieves the list of grades from the grades table in the database and populates the grade drop down lists.
* The system populates the fields with the data to be edited based on the student id passed in.

When the user clicks the save button the following processes should occur:

* Ensure no field are left blank. If any fields have been left blank the alert the user and highlight the blank fields with an asterisk.
* Validate e-mail.
* Once all fields are validated update the users and studentgrades table in the database with the selected students information and return the user to the manage students page.

## Manage Groups

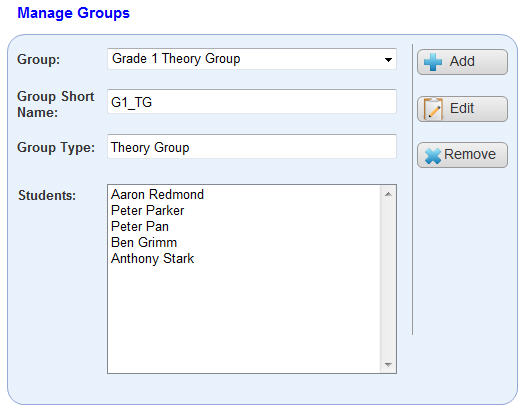


Figure 20: Manage Groups

From this page the user can view existing groups. They may also add a new group or edit an existing one.

### Links

This page directly links to the following pages:

* Add Group.
* Edit Group.

### Data

Group

* This is a list of currently available groups.

Group Short Name:

* This is a shortened version of the group name.

Group Type

* This is the type of group.

Students

* This is the list of students currently associated with the selected group.

### Processes

Add

* The system navigates to the add group page, where the user can add one or more additional groups.

Edit

* The system navigates to the edit group page, where the user can edit the currently selected group.

Remove

* Remove an existing group.

### Pseudo Code

When the page loads, the following process should take place:

* The system retrieves the lists of groups from the groups table in the database and populates the group list.
* The first group in the list is automatically selected.
* The short name and group type data fields are filled with the correct data for the selected group.
* The system retrieves all the students in this group from the groupparticipants table in the database and populates the students list.

When a different group is selected, the following processes should take place:

* The short name and group type data fields are filled with the correct data for the selected group.
* The system retrieves all the students in this group from the groupparticipants table in the database and populates the students list.

When the user clicks the remove button, the following processes should take place:

* The system prompts the user “Do you really wish to delete this group? Yes/No”.
* If the user clicks yes then the system removes the selected group from the groups table in the database and updates the group list.
* If the user clicks no then take no action.

**Note**: Removing a group does not remove students, just their association with that group.

## Add Group

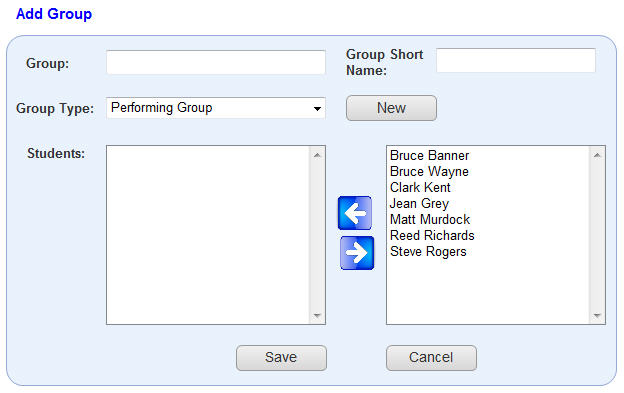


Figure 21: Add Group

If the user clicks the add group button on the manage groups page it will bring them to this page, the add group page.

### Links

This page links directly back to the following page:

* Manage Groups
* Add Group Type

### Data

Group

* This is the name of the new group the user wishes to create.

Group Short Name:

* This is a shortened version of the new group name.

Group Type

* This is the list of available group types.

Students

* This is the list of students participating in the group.

Student Master List

* This is the master list of students.

### Processes

Save

* When the user clicks this button the system saves the new group and returns the user to the manage groups page.

Cancel

* When the user clicks this button the system cancels the additional and returns the user to the manage groups page.

New

* When the user clicks this button the system navigates to the add group type page, where the user can add a new group type.

Add Student

* The selected student is added to the students list.

Remove Student

* The selected student is removed from the students list.

### Pseudo Code

As the page is loading the following processes should occur:

* The system retrieves the lists of students from the users table in the database and populates the student master list.
* The first student in the list is automatically selected.
* The system retrieves the lists of group types from the grouptype table in the database and populates the group type list.
* The first group type in the list is automatically selected.

When the user clicks the add student button, the following process should occur:

* The selected student is added to students.
* The system removes the students from the available students in the student master list.

When the user clicks the remove student button the following process should occur:

* The selected student is removed from students.
* The system adds the student into the available students in the student master list.
* If the students data field is empty then no action is taken.

When the user clicks the save button the following processes should occur:

* Ensure no field are left blank. If any fields have been left blank the alert the user and highlight the blank fields with an asterisk.
* Once all fields are validated add the new group to the groups table in the database and return the user to the manage groups page.

## Edit Group

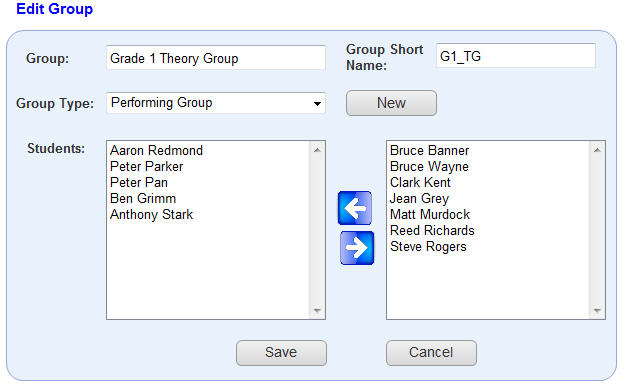


Figure 22: Edit Group

If the user clicks the edit group button on the manage groups page it will bring them to this page, the edit group page.

### Links

This page links directly back to the following page:

* Manage Groups
* Add Group Type

### Data

Group

* This is the name of the group the user wishes to edit.

Group Short Name:

* This is a shortened version of the group name to be edited.

Group Type

* This is the type of group.

Students

* This is the list of students participating in the group.

Student Master List

* This is the master list of students.

### Processes

Save

* When the user clicks this button the system saves the edited group and returns the user to the manage groups page.

Cancel

* When the user clicks this button the system cancels the edit and returns the user to the manage groups page.

New

* When the user clicks this button the system navigates to the add group type page, where the user can add a new group type.

Add Student

* The selected student is added to the students list.

Remove Student

* The selected student is removed from the students list.

### Pseudo Code

As the page is loading the following processes should occur:

* The system populates the fields with the data to be edited.
* The system retrieves the lists of students from the users table in the database and populates the student master list.
* The system removes any students from the student master list that also exists in students list.
* The first student in the master list is automatically selected.
* The system retrieves the lists of group types from the grouptype table in the database and populates the group type list.
* The system selects the currently associated group type.

When the user clicks the add student button, the following process should occur:

* The selected student is added to students.
* The system removes the students from the available students in the student master list.

When the user clicks the remove student button the following process should occur:

* The selected student is removed from students..
* The system adds the student into the available students in the student master list.
* If the students data field is empty then no action is taken.

When the user clicks the save button the following processes should occur:

* Ensure no field are left blank. If any fields have been left blank the alert the user and highlight the blank fields with an asterisk.
* Once all fields are validated update the edited group in the group table in the database and return the user to the manage groups page.

## Add Group Type



Figure 23: Add Group Type

If the user clicks the add button on either the add group page or the edit group page it will bring them to this page, the add group type page.

### Links

This page links directly back to one of the following pages, depending on the calling page:

* Add Group
* Edit Group

### Data

New Group Type

* This is the new group type.

### Processes

Save

* When the user clicks this button, the system saves the new group type and returns the user to the calling page, either add or edit group.

Cancel

* When the user clicks this button, the system cancels the new group type and returns the user to the calling page, either add or edit group.

### Pseudo Code

When the user clicks the save button, the following process should occur:

* The system checks the database to ensure the uniqueness of the new group type. If it is unique add it to the grouptype table in the database and return the user to the calling page, either edit or add class.
* If the new group type is not unique or blank, alert the user.

## Manage Lessons

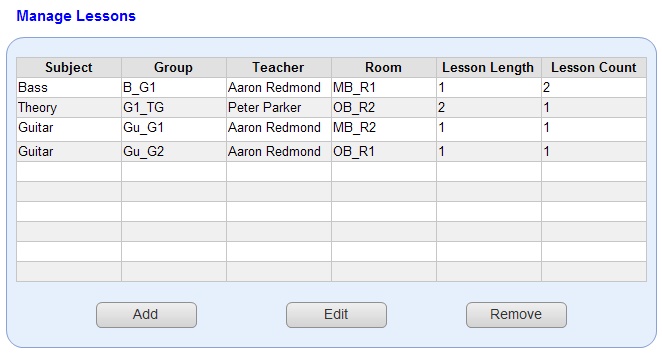


Figure 24: Manage Lessons

From this page the user can view created lessons, remove created lessons. They also have the option of adding additional lessons or editing existing ones.

### Links

This page directly links to the following pages:

* Add Lesson
* Edit Lesson

### Processes

Add

* The system navigates to the add lesson page, where the user can add one or more additional lessons.

Edit

* The system navigates to the edit lesson page, and loads the data for the selected lesson.

Remove

* Remove an existing lesson.

### Pseudo Code

As the page is loading the following process should take place:

* The system retrieves all lessons from the lessons table in the database and populates the lessons table.

When the user clicks the remove button, the following processes should take place:

* The system prompts the user “Do you really wish to delete this lesson? Yes/No”.
* If the user clicks yes then the system removes the selected lesson from the lessons table in the database and updates the lesson list.
* If the user clicks no then take no action.

## Add Lesson

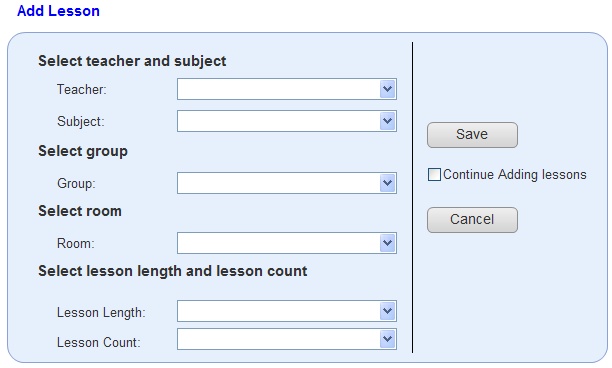


Figure 25: Add Lesson

If the user clicks the edit lesson button on the manage lessons page it will bring them to this page, the add lesson page.

### Links

This page links directly back to the following page:

* Manage Lessons

### Data

Teacher

* A list of available teachers.

Subject

* A list of available subjects.

Group

* A list of available groups.

Room

* A list of available rooms.

Lesson Length

* The length of the lesson, 1 period, 2 periods, etc.

Lesson Count

* The amount of time this lesson will be taught per week.

### Processes

Save

* Saves the new lesson and returns the user to the manage lessons page.

Cancel

* Cancels the new lessons and returns the user to the manage lessons page.

### Pseudo Code

As the page is loading the following process should take place:

* The system retrieves the list of teachers from the users table in the database and populates the teacher list.
* The system retrieves the list of subjects from the subjects table in the database and populates the subject list.
* The system retrieves the list of groups from the groups table in the database and populates the group list.
* The system retrieves the list of rooms from the rooms table in the database and populates the room list.
* The system populates the lesson length list, with the values 1-9.
* The system populates the lesson count list with values 1-9.

When the user clicks the save button, the following processes should take place:

* The system validates the data.
* The system adds the new lesson to the lessons table in the database.

When the user clicks the cancel button, the following processes should take place:

* The system discards the new lessons and navigates to the manage lessons page.

## Edit Lesson

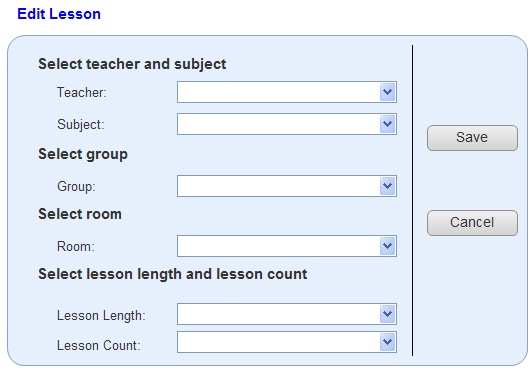


Figure 26: Edit Lesson

If the user clicks the edit lesson button on the manage lessons page it will bring them to this page, the edit lessons page.

### Links

This page links directly back to the following page:

* Manage Lessons

### Data

Teacher

* A list of available teachers.

Subject

* A list of available subjects.

Group

* A list of available groups.

Room

* A list of available rooms.

Lesson Length

* The length of the lesson, 1 period, 2 periods, etc.

Lesson Count

* The amount of time this lesson will be taught per week.

### Processes

Save

* Saves the edited lesson and returns the user to the manage lessons page.

Cancel

* Cancels the edit and returns the user to the manage lessons page.

### Pseudo Code

As the page is loading the following process should take place:

* The system retrieves the list of teachers from the users table in the database, populates the teacher list and selects the correct teacher.
* The system retrieves the list of subjects from the subjects table in the database, populates the subject list and selects the correct subject.
* The system retrieves the list of groups from the groups table in the database, populates the group list and selects the correct group.
* The system retrieves the list of rooms from the rooms table in the database, populates the room list and selects the correct room
* The system populates the lesson length list, and selects the correct amount.
* The system populates the lesson count list, and selects the correct amount.

When the user clicks the save button, the following processes should take place:

* The system validates the data.
* The system updates the edited lesson in the lessons table in the database.

When the user clicks the cancel button, the following processes should take place:

* The system discards the edit and navigates to the manage lessons page.

## Timetabling

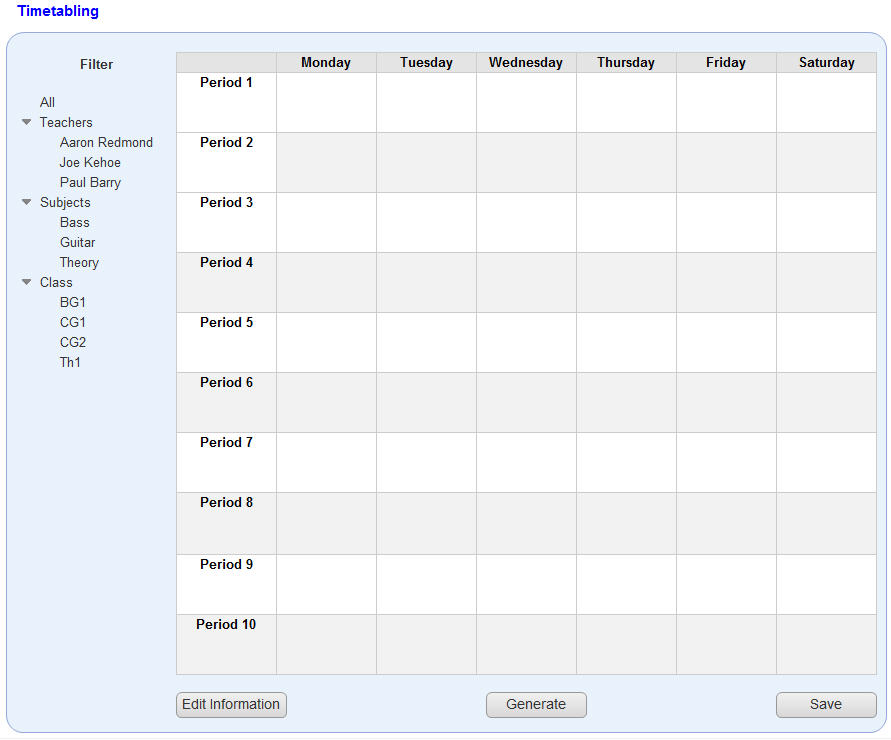


Figure 27: Timetabling

From this page the user can generate a timetable, edit timetable information and save the timetable.

### Links

This page links directly to the following page:

* Timetable Information

### Data

All timetable data on this page will be generated when the user generates a timetable.

Filter

* This option will allow the user to select how they view the timetable, room view, student view, teacher view, etc.

### Processes

Edit Information

* Navigates to the Timetable information page.

Generate

* Generates the timetable.

Save

* Saves the current timetable.

### Pseudo Code

When the user clicks the generate button, the following processes should take place:

* The system runs the generate timetable process. See appendix A for more details on the timetable generation process.
* The system displays the generated timetable.

When the user clicks the save button, the following processes should take place:

* The system presents a Save As dialog box.
* The user selects the type and the system saves the timetable in that format.

When the user clicks on a filter option, the system should:

* Display the correct timetable for the selected option.

## Timetabling Information

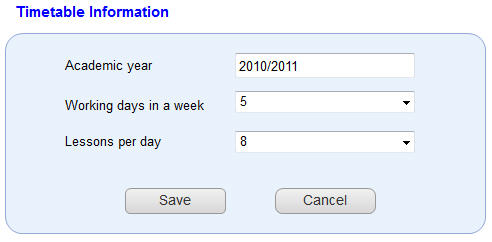


Figure 28: Timetable Information

From this page the user can edit the information associated with the timetable.

### Data

Academic year

* + This is the current academic, 2010/2011, 2011/2012, etc.

Working Days in a week

* + The number of days per week that the school will be running lessons. This also controls the number of columns in the timetable.

Lessons per day

* + The number of periods that the school will have per day. This controls the number of rows in the timetable.

### Processes

Save

* + Saves the edited timetable information and returns the user to the timetabling page.

Cancel

* + Discards any changes made to the timetable information and returns the user to the timetabling page.

### Pseudo Code

As the page loads the following process should take place:

* The system retrieves the timetable information from the timetableinfo table in the database and populates the data fields.

When the user clicks the save button, the following processes should take place:

* Check to ensure that the academic year field is not blank. If it is blank prompt user.
* Update the timetableinfo table in the database with edited information.

## Instruments

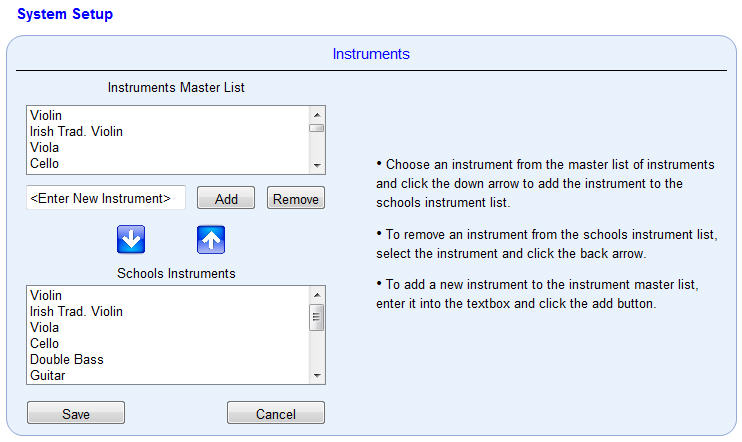


Figure 29: System Setup - Instruments

This page is for the setup of instruments both used and taught by the school. It may be accessed from the system setup section of the menu.

### Links

This page links directly back to the following page:

* Admin Home

### Data

Instrument Master List

* This is the master list of instrument available to the school.

New Instrument

* This is the new instrument to be added to the instrument master list.

School Instruments List

* This is the list of instruments currently being used by the school.

### Processes

Add

* Clicking this button adds the new instrument to the instrument master list.

Remove

* Clicking this button removes the selected instrument from the instrument master list.

Add Instrument

* Clicking this button adds the instrument selected in the instrument master list to the school instruments list.

Remove Instrument

* Clicking this button removes the instrument selected in the school instruments list.

Save

* Clicking this button saves both the instrument lists.

Cancel

* Clicking this button cancels any changes made.

### Pseudo Code

As this page loads, the following process should take place:

* The system populates the instrument master list from the instruments table in the database with the instruments which are not currently being used.
* The system populates the school instruments list from the instruments table in the database with the instruments which are being used

When the user clicks the add button, the following processes should take place:

* The system ensures that the new instrument is unique and not blank.
* The system adds the new instrument to the instrument master list and updates the instruments table in the database.

When the user clicks the remove button, the following processes should take place:

* The system prompts the user “are you sure you want to remove this instrument? Yes/No”, if the user replies yes then remove the instrument from the instrument master list and update the instruments table in the database.

When the user clicks the add instrument button, the following processes should take place:

* The system adds the selected instrument to the school instruments list.
* The system removes the selected instrument from the instrument master list.

When the user clicks the remove instrument button, the following processes should take place:

* The system removes the selected instrument from the school instruments list.
* The system adds the selected instrument to the instrument master list.

When the user clicks the save button, the following processes should take place:

* The system saves both the instrument master list and the school instruments list and returns the user to the admin home page.

When the user clicks the cancel button, the following processes should take place:

* The system cancels any edit since the last save and returns the user to the admin home page.

## Personalisation

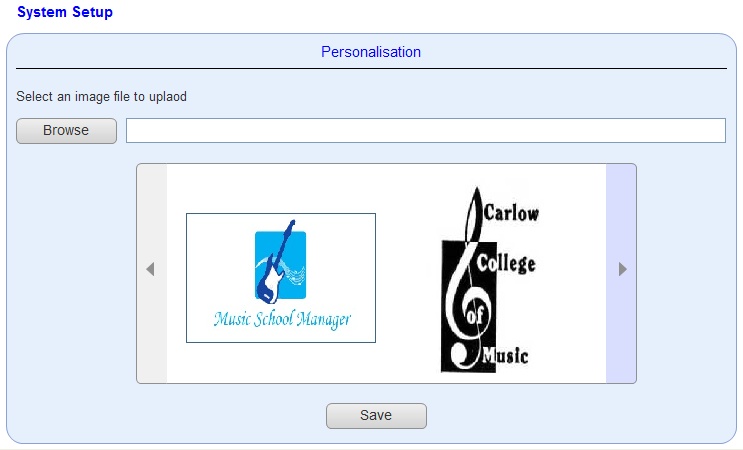


Figure 30: System Setup - Personalisation

This page is to allow the personalisation of the website. It will allow an administrator to upload and select an image which can then be used as a logo throughout the site. This page can be accessed through the system setup section of the menu.

### Links

The page links directly back to the following page:

* Admin Home

### Data

File Name

* This is the user selected file for upload.

Image List

* This is the list of images the user can choose form.

### Processes

Browse

* Clicking this button displays a dialog which allows the user to choose a file for upload.

Save

* Clicking this button saves the new personalisation settings and returns the user to the admin home page.

### Pseudo Code

AS this page loads, the following processes should occur:

* The system populates the image list with the images currently stored in the personalisation table in the database.

When the user clicks the browse button, the following processes should occur:

* The system displays a browse file dialog box, which allows the user to select a file for upload.
* The type of files the user can select should be restricted to certain image types, such as jpg and bmp.
* If the user clicks ok the selected file is uploaded to the images directory and a reference added to the personalisation table in the database.
* The system refreshes the image list.
* If the user clicks cancel then no action takes place.

When the user clicks the save button, the following processes should occur:

* The system sets the chosen image as the image to be used on the website, and returns the user to the admin home page.

## News

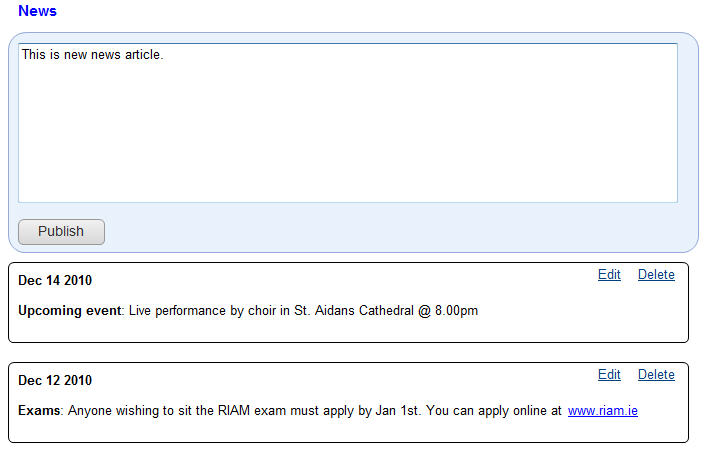


Figure 31: News

This page allows the admin to publish, edit or delete a news item. This news page can be accessed from the system setup in the main menu.

### Data

Publish Item

* This is the news item to be published.

News Item

* These are published news items.

### Processes

Publish

* When the user clicks this button the publish item is published.

Edit

* When the user clicks this link, the news item they wish to edit is displayed in the publish item field.

Delete

* When the user clicks this button, the selected news item is deleted.

### Pseudo Code

As this page loads, the following processes should occur:

* The system retrieves the latest new items from the news table in the database.

When the publish button gets clicked, the following processes should occur:

* The system ensures the publish item field is not blank, if it is then alert the user.
* The system publishes the new news item, and updated the news table in the database.

When the edit link is clicked, the following processes should occur:

* The system places the content of the selected news item into the publish item field.

When the delete link is clicked, the following processes should occur:

* The system prompts the user “Delete this article? Yes/No”, if yes then delete the article from the news table in the database and reload the news items.
* If no then no action is taken.

## My Account

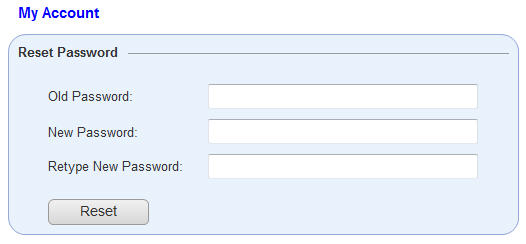


Figure 32: My Account

This page allows an admin to change setting for their account. This page can be accessed from the My Account link in the User Panel section of the menu.

At the moment the only option available is the reset password option. Any future functionality regarding an admin account will be placed in here.

### Data

Old Password

* This is the current password, which the user wishes to change.

New Password

* This is the new password which the user wishes to use.

Retype New Password

* This is a copy of the new password the user wishes to use.

### Processes

Reset

* When this button is clicked the password is changed.

### Pseudo Code

When the user clicks the reset button, the following processes occur:

* The system ensures that none of the fields have been left blank, if they are then mark the fields with an asterisk and alert the user.
* The system ensures that the old password matches the password currently in the database, if it does not match then alert the user.
* The system ensures that the new password and retype new password fields match, if they do not match then alert the user.
* The system updates the password in the users table in the database.

# Phase 3 – Teacher

Phase 3 is involved with implementing the teacher side of the application. The teacher section is rather small in comparison to the admin section.

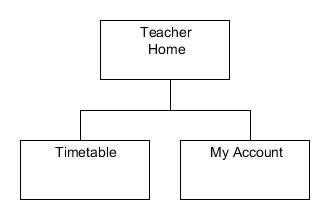


Figure 33: Phase 3

## Home

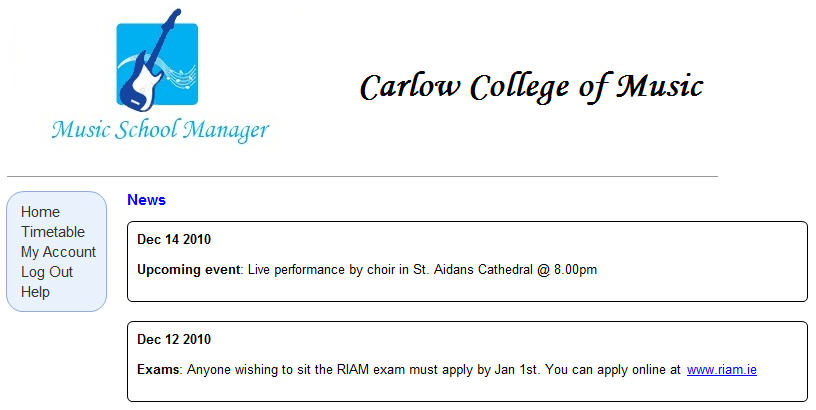


Figure 34: Teacher Home

This is the first page a teacher will see when they successfully log in. From here they can access all of the functionality available to a teacher.

All the latest news can also be viewed on the teacher’s home page.

### Links

This page directly links to the following pages:

* Timetable
* My Account

### Pseudo Code

As this page loads, the following process should occur:

* The system retrieves the latest new items from the news table in the database.

## Timetable

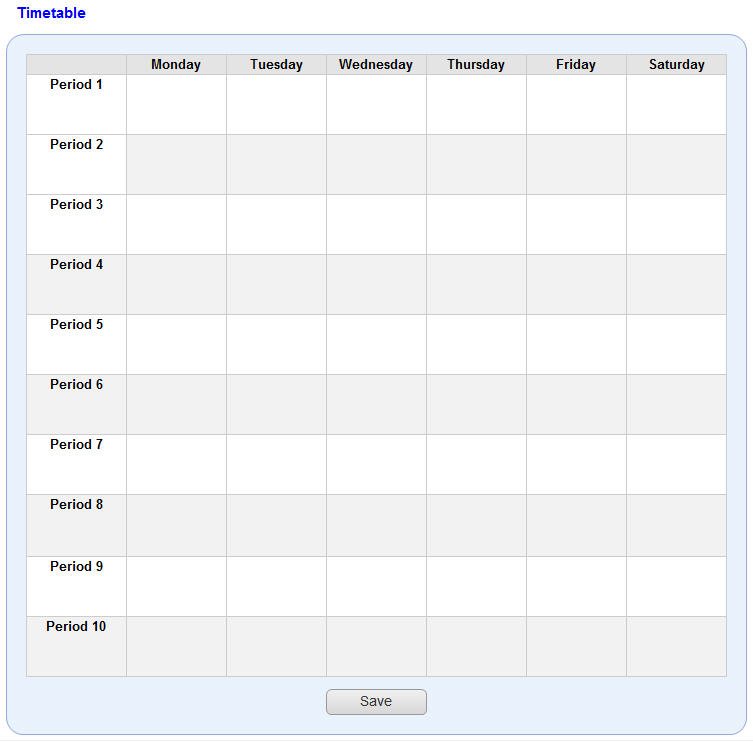


Figure 35: Teacher Timetable

From the page the teacher can view their personal timetable. They may also save the timetable.

### Processes

Save

* The user can click this button to save their timetable.

### Pseudo Code

As this page loads, the following process should occur:

* The system loads the personal timetable for the teacher.

When the user clicks the save button, the following processes should take place:

* The system presents a Save As dialog box.
* The user selects the type and the system saves the timetable in that format.

## My Account

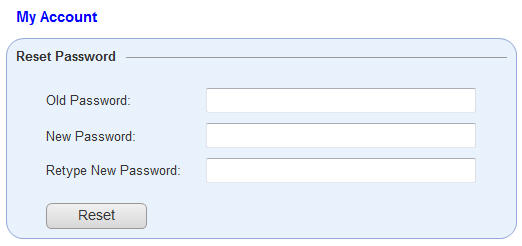


Figure 36: My Account

This page behaves exactly as described in section3.28.

# Phase 4 – Student

Phase 4 is involved with implementing the student side of the application. The student section is very similar to the teacher section and therefore should be the easiest to implement.

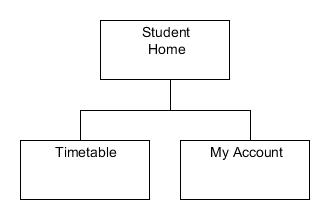
****

Figure 37: Phase 4

## Student Home

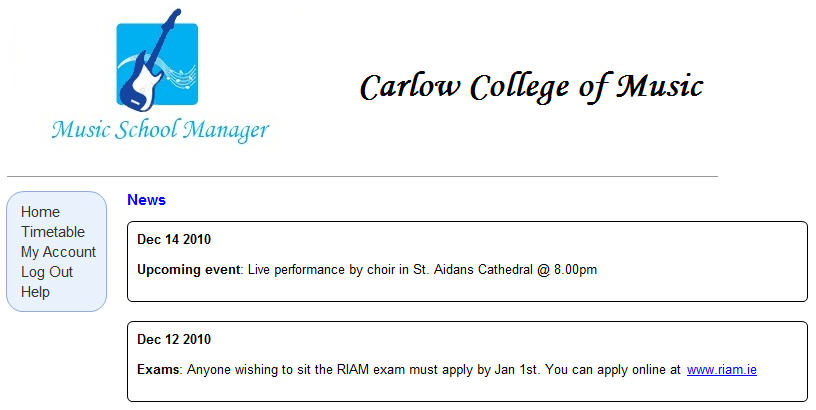


Figure 38: Student Home

This is the first page a student will see when they successfully log in. From here they can access all of the functionality available to a student.

All the latest news can also be viewed on the student’s home page.

### Links

This page directly links to the following pages:

* Timetable
* My Account

### Pseudo Code

As this page loads, the following process should occur:

* The system retrieves the latest news items from the news table in the database.
* This process is exactly the same as the one described in section 4.1.2.

## Timetable

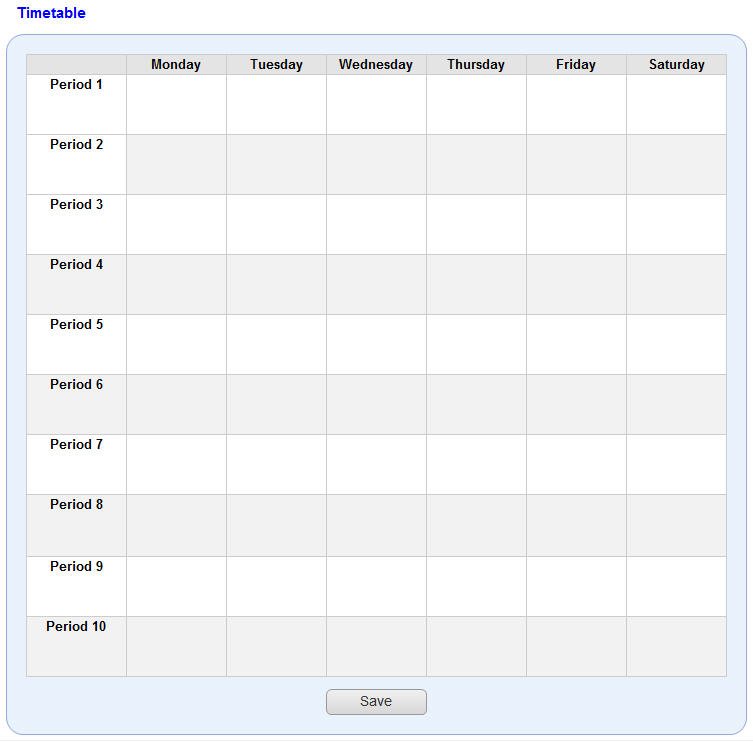


Figure 39: Student Timetable

From the page the student can view their personal timetable. They may also save the timetable.

### Processes

See section 4.2.1.

### Pseudo Code

See section 4.2.2.

## My Account

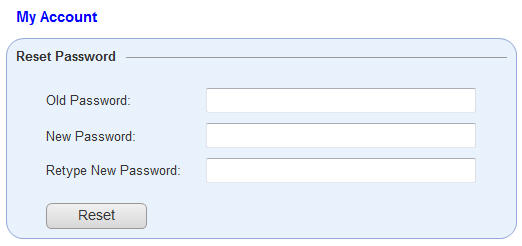


Figure 40: My Account

This page behaves exactly as described in section3.28.

# Appendix A – Timetable Generation

## A little bit on genetic algorithms

Currently the timetable for the music school is constructed manually or in a spreadsheet package such as Microsoft excel. This can be an arduous and time consuming process as the person compiling the timetable has to try and schedule every lesson while trying to meet all the restraints. Restraints can be categorised into two sets: hard constraints and soft constraints, with hard constraints having a higher priority than soft constraints.

**Hard Constraints**

A hard constraint has to be satisfied in order to complete a working timetable. The following list contains some of the hard constraints:

* A room must not be double booked.
* The number of classes allowed simultaneously is restricted to the number of rooms available.
* Teachers must not be double booked.
* All lessons must be scheduled.

**Soft Constraints**

A soft constraint should be satisfied wherever possible; however it is not necessary for the completion of the timetable. A good example of a soft constraint is:

* Teacher X would like the following period free; 10am to 11am on Mondays.
* Classes should be distributed evenly over the week.

A computer algorithm should help speed up this process and meet all hard restraints. The algorithm which we will be implementing to solve the timetabling will be a genetic algorithm. The general form for a genetic algorithm is defined as follows:

|  |
| --- |
| procedure genetic algorithm;  begin  set time t := 0;  initialise the population P(t);  while termination conditions not met do  begin  evaluate the fitness of each member of the population P(t);  select members from population P(t) based on fitness;  produce the offspring of these pairs using genetic operators;  replace, based on fitness, candidates of P(t), with these offspring;  set time t := t + 1;  end  end |

Table 1: Genetic Algorithm [1]

## Implementation

There are several ways in which we can implement a genetic algorithm, the first of which is to wrap existing C++ classes in a PHP extension.

The existing C++ implementation of the genetic algorithm is available to download from the code project website. It is as such covered by the code project open license (CPOL), and free for us to use in this project. [2]

The C++ application takes its input from a basic text (configuration) file, which contains all the data necessary to create a timetable. The C++ application parses this file and applies the genetic algorithm to it; with a new timetable being created every pass through the algorithm. The algorithm continues to run until a specific stopping condition is met, at which point we should end up with the best possible timetable.

In order for us to implement this C++ application, we must wrap it in a PHP extension. This is done by:

* Instructing the PHP build system to use a C++ compiler. This is done by calling the PHP\_REQUIRE\_CXX() macro in your config.m4 file.

|  |
| --- |
| PHP\_ARG\_ENABLE(vehicles,  [Whether to enable the "vehicles" extension],  [ --enable-vehicles Enable "vehicles" extension support])  if test $PHP\_VEHICLES != "no"; then  PHP\_REQUIRE\_CXX()  PHP\_SUBST(VEHICLES\_SHARED\_LIBADD)  PHP\_ADD\_LIBRARY(stdc++, 1, VEHICLES\_SHARED\_LIBADD)  PHP\_NEW\_EXTENSION(vehicles, vehicles.cc car.cc, $ext\_shared)  Fi |

Figure 41: Example Code showing PHP\_REQUIRE usage [3]

* Creating a PHP header file which will contain references used by PHP to load the extension.
* Creating a configuration file to prepare the extension for compiling.
* Creating a PHP class for every C++ class. The new PHP class must have identical (or similar) definitions.
* Connecting the PHP class to the C++ class, each instance of the PHP class must have access to exactly one instance of the C++ class.
* Creating objects which allow us to call the functionality of the C++ classes.

This is just a brief overview, with the full tutorial available online [3]. There is also a development tool called SWIG which allows a developer connect programs written in C++ and other languages with a variety of high-level programming languages such as PHP, Python, etc. [4]

The second method available to us is an implementation of the genetic algorithm written in PHP. This implementation is available from the PHPclasses website. [5]

This algorithm provides us with a base class with which to implement a genetic algorithm, allowing us to perform crossover and mutation operations. For us to implement this, we just need to create a class to represent a timetable and pass it to the algorithm.

We could also use a genetic algorithm library written in Java. There are two available: Jenetics and JGAP. Both these libraries are very comprehensive and contain good examples. [6][7]

# References

1. Luger, George F., 2005, Artificial intelligence (5thed.): structures and strategies for complex problem-solving, Addison-Wesley
2. Making a Class Schedule Using a Genetic Algorithm, 2008, The Code Project, available: <http://www.codeproject.com/KB/recipes/GaClassSchedule.aspx> [accessed 27 January 2011]
3. Wrapping C++ Classes in a PHP Extension, 2009, Zend Developer Zone, available: <http://devzone.zend.com/article/4486> [accessed 27 January 2011]
4. SWIG, 2009, SWIG, available: <http://www.swig.org/index.php> [accessed 27 January 2011]
5. Generic genetic algorithms base implementation, 2009, Rafael Pinto, available: <http://www.phpclasses.org/package/2567-PHP-Generic-genetic-algorithms-base-implementation.html> [accessed 27 January 2011]
6. Jenetics — Java Genetic Algorithm, 2011, Jenetics, available: <http://jenetics.sourceforge.net/index.html> [accessed 27 January 2011]
7. JGAP – Java Genetic Algorithms Package, 2011, JGAP, available: <http://jgap.sourceforge.net/index.html> [accessed 27 January 2011]